Vom Fachbereich Informatik der Technischen Universität Kaiserslautern zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation. Datum der wissenschaftlichen Aussprache: 02.05.2012

Information Extraction on the Semantic Web

Utilizing the Resource Description Framework in Information Extraction

Benjamin Adrian

June 17, 2012

Promotionskommittee:

Prof. Dr. Andreas Dengel, Technische Universität Kaiserslautern,
Erstgutachter

Prof. Dr. Philipp Cimiano, Universität Bielefeld,
Zweitgutachter

Prof. Dr. Arnd Poetzsch-Heffter, Technische Universität Kaiserslautern,
Vorsitzender, Dekan

D 386

In January 11, 2012, this document was submitted as doctoral thesis to the Department of Computer Science, University of Kaiserslautern. In May 02, 2012, this dissertation was defended successfully by Benjamin Adrian.

This document was set in Times New Roman by the author using KOMA-Script and LaTeX.

Hauptstr. 42,
D-67731 Otterbach, Germany
benjamin.horak@gmail.com

Acknowledgments

Although by nature a doctoral thesis has to be performed by the author on his own, he would never bring it to success without the help of others.

Fortunately, while investigating on "Information Extraction on the Semantic Web" I had the opportunity to collaborate with a number of outstanding and supporting colleagues and researchers.

First, I want to thank my adviser Andreas Dengel. Since the beginning of my research, he confirmed me the importance of the topic of this work and encouraged me to proceed. Andreas made it possible for me to discuss and present emerging research objectives and results at international conferences. This comprised the doctoral consortium at the International Semantic Web Conference 2009 in Washington DC. There, I had the chance to meet Tim-Berners Lee, James Hendler, Ivan Herman, and all the other folks of the Semantic Web community. Andreas also enabled me contributing to two working groups of the World Wide Consortium, namely the RDFa WG and the RDF Web application WG. These activities helped a lot in understanding the research gap between Information Extraction and the Semantic Web. Especially in the beginning of my research, my close collaboration with Andreas laid the foundation of my career as researcher.

I would like to say a big thank you to Marcus Liwicki. Writing my thesis, Marcus supported me with excellent remarks and hints on structuring and editing this document. His thoroughness in willingness for understanding every bit of this work was a real support for me.

I want to thank my office mate Jörn Hees for his patience. Jörn always had the time for brainstorming and discussing topics and problems. He also has the talent to ask the right questions at the right time.

I like to thank Thomas Roth-Berghofer for his excellent advises. During my work time at DFKI, Thomas has become a very good friend of mine.

I had the chance to work with excellent students on topics of this thesis, namely Sebastian Ebert and Sebastian Sperber. Collaborating with them was always very inspiring.

In general, thank you Knowledge Management Group at DFKI. It was great fun working with you for the last five years.

Last but definitely not least, I am deeply grateful to my wife Denise, my children Emma and Max, and my family. They were a great support in recent years. Without them it would not have been possible to finishing this doctoral thesis successfully.

Abstract

Dealing with information in modern times involves users to cope with hundreds of thousands of documents, such as articles, emails, Web pages, or News feeds. Above all information sources, the World Wide Web presents information seekers with great challenges. It offers more text in natural language than one is capable to read.

The key idea for this research intends to provide users with adaptable filtering techniques, supporting them in filtering out the specific information items they need. Its realization focuses on developing an Information Extraction system, which adapts to a domain of concern, by interpreting the contained formalized knowledge.

Utilizing the Resource Description Framework (RDF), which is the Semantic Web's formal language for exchanging information, allows extending information extractors to incorporate the given domain knowledge. Because of this, formal information items from the RDF source can be recognized in the text. The application of RDF allows a further investigation of operations on recognized information items, such as disambiguating and rating the relevance of these. Switching between different RDF sources allows changing the application scope of the Information Extraction system from one domain of concern to another. An RDF-based Information Extraction system can be triggered to extract specific kinds of information entities by providing it with formal RDF queries in terms of the SPARQL query language. Representing extracted information in RDF extends the coverage of the Semantic Web's information degree and provides a formal view on a text from the perspective of the RDF source.

In detail, this work presents the extension of existing Information Extraction approaches by incorporating the graph-based nature of RDF. Hereby, the pre-processing of RDF sources allows extracting statistical information models dedicated to support specific information extractors. These information extractors refine standard extraction tasks, such as the Named Entity Recognition, by using the information provided by the pre-processed models. The post-processing of extracted information items enables representing these results in RDF format or lists, which can now be ranked or filtered by relevance. Post-processing also comprises the enrichment of originating natural language text sources with extracted information items by using annotations in RDFa format.

The results of this research extend the state-of-the-art of the Semantic Web. This work contributes approaches for computing customizable and adaptable RDF views on the natural language content of Web pages. Finally, due to the formal nature of RDF, machines can interpret these views allowing developers to process the contained information in a variety of applications.

About the author

Personal Data

Benjamin Adrian

Education

2006	German Diplom in Computer Science
2001 - 2006	Studied Applied Computer Science at University of Kaiserslautern, Germany
1991 - 2000	Helmholtz-Gymnasium, 66482 Zweibrücken, Germany

Experience

2006 - 2011	Researcher, German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany
2010 - 2011	Member of the RDFa Working Group, World Wide Web Consortium (W3C)
2005 - 2006	Undergraduate research assistant, German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany
2005	6 month internship at SAP, St. Ingbert, Germany
2004 - 2005	Undergraduate research assistant, University of Kaiserslautern, Dept. Database and Information Systems, Kaiserslautern, Germany
2001 - 2003	Undergraduate student assistant, Business and Innovation Center (BIC GmbH), Kaiserslautern, Germany

List of Abbreviations

AI	Artificial Intelligence
CBD	Concise Bound Description
ConLL	Conference on Computational Natural Language Learning
CRF	Conditional Random Field
EBNF	Extended Bachus-Naur-Form
FSM	Finite-state Machine
FOAF	Friend of a Friend
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IE	Information Extraction
IR	Information Retrieval
LOD	Linking Open Data
MUC	Message Understanding Conference
NER	Named Entity Recognition
NLP	Natural Language Processing
OBIE	Ontology-based Information Extraction
OWL	Web Ontology Language
PCA	Principle Component Analysis
POS	Part-of-speech
RDFa	RDF in Attributes
RDF	Resource Description Framework
RDFS	RDF Vocabulary Description Language
SCOOBIE	Service Ontology-based Information Extraction
SGML	Standard Generalized Markup Language
SPARQL	SPARQL Query Language
SQL	Structured Query Language
URI	Uniform Resource Identifier
URL	Unique Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

Contents

1 Introduction **21**
- 1.1 Motivation . . . 22
- 1.2 Hypotheses . . . 23
- 1.3 Contributions . . . 24
- 1.4 Prerequisites . . . 25
- 1.5 Terminology . . . 26
- 1.6 Outline . . . 27

2 Computation on natural language **29**
- 2.1 Processing natural language . . . 30
  - 2.1.1 Syntactics . . . 31
  - 2.1.2 Semantics . . . 32
  - 2.1.3 Pragmatics . . . 33
- 2.2 Information Extraction . . . 33
  - 2.2.1 Historic outline . . . 35
  - 2.2.2 State-of-the-Art . . . 38
- 2.3 Unsolved and emerging challenges . . . 48
- 2.4 Contributions . . . 50

3 Representing knowledge on the Web **51**
- 3.1 Computing with semantics in the Semantic Web . . . 52
- 3.2 Resource Description Framework . . . 53
- 3.3 Representing instance knowledge . . . 55
- 3.4 Representing properties . . . 56
- 3.5 Representing classes . . . 57
- 3.6 Representing multiple RDF graphs . . . 58
- 3.7 Vocabularies . . . 58
- 3.8 Querying with SPARQL . . . 59
- 3.9 Utilizing RDF from the Web . . . 61
- 3.10 Unsolved and emerging challenges . . . 62
- 3.11 Contributions . . . 63

4 Ontology-based Information Extraction **65**
- 4.1 Ontologies in Information Extraction . . . 66

4.2 Ontology-based Information Extraction Systems . . . 69
4.3 Extraction ontologies . . . 70
4.4 Unsolved and emerging challenges . . . 72
4.5 Contributions . . . 72

5 Foundations for utilizing RDF in Information Extraction 75
5.1 Linking URIs and textual references . . . 76
5.2 RDF components . . . 77
5.2.1 Literals . . . 77
5.2.2 Datatype properties . . . 77
5.2.3 Types . . . 79
5.2.4 Instances . . . 80
5.2.5 Object properties . . . 81
5.3 Semantic entity recognition process . . . 81
5.4 Required technological fundamentals . . . 85
5.4.1 Suffix arrays . . . 85
5.4.2 Matrix computations . . . 86
5.4.3 Hierarchical clustering . . . 87
5.4.4 Descriptive statistics . . . 88
5.4.5 Principle component analysis . . . 90
5.4.6 Link analysis in graphs . . . 91
5.4.7 Maximum entropy models . . . 93
5.4.8 Conditional Random Field . . . 94
5.5 Summary and Conclusion . . . 95
5.5.1 Summary . . . 95
5.5.2 Conclusion . . . 96

6 Preprocessing feature descriptions from text and RDF graphs 97
6.1 Features in Information Extraction . . . 98
6.2 A relational model of RDF data . . . 99
6.3 Clustering correlating classes in RDF graphs . . . 102
6.3.1 Hierarchical clustering . . . 105
6.3.2 Principle Component Analysis . . . 108
6.4 RDF graph statistics . . . 109
6.4.1 Usage statistics of datatype properties . . . 109
6.4.2 Estimating cardinalities . . . 111
6.4.3 Estimating characteristic relations between classes . . . 113
6.5 Text corpus statistics . . . 114
6.5.1 Term and document frequencies . . . 114
6.5.2 Combining text corpus with RDF graph statistics . . . 115
6.6 Mining datatype properties for proper names . . . 115

6.7 Aligning datatype properties with regular expressions . . . 117
6.8 Automatically labeling a text corpus with classes . . . 118
6.9 Experiments and evaluations . . . 118
6.9.1 Datasets . . . 118
6.9.2 Clustering classes in DBpedia . . . 120
6.9.3 Proper name mining in DBpedia . . . 125
6.9.4 Learning a Markov chain from DBpedia . . . 128
6.9.5 Matching regular expressions with DBpedia literals . . . 129
6.9.6 Labeling the ConLL 2003 Corpus . . . 131
6.10 Summary and conclusion . . . 131
6.10.1 Summary . . . 131
6.10.2 Conclusion . . . 132

7 Processing the Semantic Entity Recognition 133
7.1 Information Extractors . . . 134
7.2 Filtering text for proper names . . . 135
7.3 Spotting text for datatype property values . . . 137
7.4 Linking named entities to formal instances . . . 140
7.5 Resolving ambiguous semantic entities . . . 142
7.6 Rating relevance of semantic entities in text . . . 146
7.7 Classifying semantic entities . . . 149
7.8 Predicting object properties between semantic entities . . . 150
7.9 Experiments . . . 155
7.9.1 Test Corpora . . . 156
7.9.2 Evaluation metrics . . . 157
7.9.3 Accelerating the recognition of semantic entities . . . 159
7.9.4 Naive recognition of semantic entities . . . 162
7.9.5 Filtering entity recognition by datatype properties . . . 163
7.9.6 Graph-based disambiguation . . . 166
7.9.7 Entity classification on automatically generated training data . . . 168
7.9.8 Classification-based disambiguation . . . 179
7.9.9 Ranking extraction results by relevance . . . 180
7.9.10 Predicting object properties . . . 185
7.10 Summary and Conclusion . . . 186
7.10.1 Summary . . . 186
7.10.2 Conclusion . . . 188

8 Incorporating SPARQL and RDF serializations into Information Extraction 189
8.1 Post-processing IE results . . . 190
8.2 Ranking IE results . . . 191
8.3 Serializing extraction results in RDF . . . 193

8.4 Annotating IE results in Web pages . . . 195
8.5 Filtering IE results by specifying templates in SPARQL . . . 196
8.5.1 Extracting Filtering Statements from SPARQL templates . . . 197
8.5.2 Inferring Filtering Statements from SPARQL templates . . . 198
8.6 Summary and Conclusion . . . 199
8.6.1 Summary . . . 199
8.6.2 Conclusion . . . 200

9 Applications of RDF-based Information Extraction 201
9.1 RDF-based Information Extraction Systems . . . 202
9.1.1 SCOOBIE . . . 202
9.1.2 A comparative view on Semantic Entity Recognition systems . . . 205
9.1.3 Epiphany . . . 206
9.1.4 Sterntaler . . . 208
9.2 Additional applications and experiments . . . 209
9.2.1 Labeling a digital document image . . . 209
9.2.2 Semantic Desktop . . . 210
9.2.3 Explanations . . . 211
9.2.4 Textual case-based reasoning . . . 212
9.3 Summary and Conclusion . . . 212
9.3.1 Semantic content enrichment . . . 212
9.3.2 Position . . . 214
9.3.3 Conclusion . . . 215

10 Concluding Information Extraction on the Semantic Web 217
10.1 Summary . . . 218
10.2 Discussion . . . 218
10.3 Lessons-learned . . . 219
10.4 Conclusion . . . 220
10.5 Future Work . . . 221
10.5.1 Short-term investigations . . . 221
10.5.2 Long-term investigations . . . 223
10.6 Acknowledgments . . . 224

Bibliography 225

List of Figures

1.1 Degrees of formalism. 23

2.1 Two parse tree alternatives. 32
2.2 Information Extraction template. 36
2.3 A simple extract of a finite-state machine. 39
2.4 General architecture of IE systems. 40
2.5 Components used for recognizing entities in text. 42
2.6 Feature types used for representing named entities. 44

3.1 The Semantic Web layer cake (W3C, 2007). 52
3.2 RDF graph in TURTLE syntax. 54
3.3 Graph visualization of RDF data about the "Battle of Kaiserslautern". . . 54
3.4 Symmetric Concise Bound Description of `dbp:Kaiserslautern`. 55
3.5 LOD cloud diagram. 61
3.6 Extended Semantic Web layer cake 63

4.1 Ontology learning stack, by Cimiano [2006]. 67
4.2 The OBIE system, by Wimalasuriya and Dou [2010]. 68
4.3 Extraction ontology. 71

5.1 Triadic co-relation between entity references and URI references. 76
5.2 Markup text segments as formal literal values of an RDF graph. 78
5.3 Assign RDF datatype properties to recognized entities. 79
5.4 Disambiguating word senses by explicit `rdf:type` statements. 80
5.5 Semantic link between a named entity and a URI reference. 81
5.6 RDF resources are interlinked with object properties. 82
5.7 RDF-based Information Extraction architecture. 83
5.8 Correlation values between variables in a 2-dimensional space. 90

6.1 Model creation by pre-processing RDF and corpus data. 98
6.2 Dendrogram. 106
6.3 Colored extract of a correlation matrix. 120
6.4 Taxonomic precision ratios of PCA and hierarchical clusterings. 122
6.5 PCA generated clustering of DBpedia classes. 123

6.6 Hierarchical clustering based generated clustering of DBpedia classes. . . 124
6.7 Histograms of proper name ratings. 126
6.8 Markov chains on object properties. 127

7.1 Processing steps. 134
7.2 Filtering plain text for noun phrases. 136
7.3 Like dictionaries in linguistics, datatype properties subsume literal values. 137
7.4 Algorithm for matching text with datatype property values. 138
7.5 Linking named entities in text to instances in RDF graphs. 141
7.6 Relations between recognized instances indicate resolutions to ambiguities. 143
7.7 Some recognized semantic entities are more relevant than others. 147
7.8 Instance/predicate-object matrix of triples in Example 7.10. 152
7.9 Similarity values for matrix in Figure 7.8. 153
7.10 Predicted values for matrix in Figure 7.8. 154
7.11 Sets of relevant and recognized instances. 158
7.12 Histogram of length of literals in DBpedia. 160
7.13 Count of query results in terms of prefix lengths. 161
7.14 Processing times needed for querying and comparing literals. 163
7.15 Histograms of ambiguity ratios of datatype properties. 164
7.16 Histograms of ambiguity ratios of datatype properties. 165
7.17 Accuracy of classification results on ConLL2003 test data. 170
7.18 Accuracy of classification results in a 10-fold cross validation. 171
7.19 Comparison of window lengths and n-gram conjunctions. 172
7.20 Comparison between content and context features. 173
7.21 Learning curves of maximum entropy models. 173
7.22 Impact of known semantic entities on context features. 174
7.23 Correlation between threshold and classification accuracies. 175
7.24 Pascal ontology of workshops and conferences. 176
7.25 Classifier's accuracies of predicted properties (a) and types (b). 177
7.26 Correlation between threshold and classification accuracies. 178
7.27 Precision ratios of an ideal type classifier on ambiguous entities. 179
7.28 Ranking correlations between the five test corpora. 182
7.29 Correlation heat maps between rankings. 184

8.1 The architecture of an RDF based IE system. 190
8.2 Cycle between extracting information and annotating formal knowledge. . 191
8.3 A list of recognized instances sorted by relevance. 192

9.1 Extended Semantic Web layer cake. 202
9.2 The Epiphany RDFa annotator. 207
9.3 Screenshot of the Sterntaler faceted Web search. 208

9.4 Rendering recognized semantic entities on top of a document image. . . . 209
9.5 The Nepomuk DropBox. 210
9.6 Explanations on IE results. 211

List of Tables

6.1 Correlation matrix about ambiguity, coverage, and idf. 126
6.2 Proper name ratings for common classes in DBpedia. 128
6.3 Outgoing transitions by object properties of `dbp-ont:Building`. 129

7.1 Comparison of frequencies between noun phrases and other words in text. 160
7.2 Response times of string or integer based hash values. 162
7.3 Recognized rates of longest match and all matches strategies. 162
7.4 Instance recognition results. 163
7.5 Rankings of the top three datatype properties. 166
7.6 Comparison of graph-based disambiguation algorithms. 167
7.7 DBpedia classes corresponding with ConLL2003 labels. 168
7.8 Coverage of datatype properties used classes of the DBpedia. 169
7.9 Pearson correlation coefficient matrix. 169
7.10 Confusion matrices listing two types of context features. 175
7.11 Mean average precision ratios of ranking metrics on each corpus. 181
7.12 Best rankings from all kinds of combinations of metrics. 183
7.13 Leave one out evaluation results of fact predictors. 185

9.1 A system comparison of instance recognition services. 205

1 Introduction

It's Not Information Overload. It's Filter Failure.

(Clay Shirky, 2008)

Google's software engineers Alpert and Hajaj [2008] stated that in 2008, their systems hit a milestone of handling one trillion (as in 1 000 000 000 000) Unique Resource Locators (URLs) on the Web at once. With respect to this incredible amount of published information, an information overload resulting in disorientation and lack of responsiveness may be assumed at worst. The answer of Shirky [2008], a US writer and consultant, to the social and economic effects of massive information on Web technologies is very clear: *"It's Not Information Overload. It's Filter Failure"*. He states that efficient information management bases on efficient filtering techniques. Unfortunately, these information-filtering techniques are still immature. In this work, we are following Clay Shirky's general assumption that information independent from its amount is a blessing. It investigates the application of knowledge representation techniques from the Web to creating semantic filters that will prevent users from being overloaded with information.

This chapter outlines the motivation, the hypotheses, and the main contributions of this work. In Section 1.1, Web technologies are motivated for developing efficient information filtering and extraction facilities. Section 1.2 addresses the research objective by claiming three research hypotheses. Main contributions to the Semantic Web and Information Extraction are given in Section 1.3. Section 1.4 lists necessary background to understand the scope of this work. Editorial conventions and the used terminology is outlined in Section 1.5. Finally, Section 1.6 summarizes the structure of this work.

1.1 Motivation

In 1990, Tim Berners-Lee encouraged users to publish and distribute information on a world embracing software system called the World Wide Web (WWW). The Web is spanned between distributed, independent Web servers running at sites all over the world. On such a Web server, we can deploy any kind of resource, from text documents, videos, audios, to files. To address a published resource on the Web, a Uniform Resource Identifier (URI) has to be assigned. All threads in the World Wide Web base on such URIs. They are implemented as hyperlinks that associate an originating Web resource with a target resource by referring to its URI. The Hypertext Transfer Protocol (HTTP) allows requesting URIs from within a Web browser on a client machine, which results in a transfer of information from the Web server to the browser.

By nature, the WWW is a web of documents. Its design focuses on the use of hypertext [Jacobs and Walsh, 2004]. When querying a standard Web search engine, the common stereotype of a returned Web resource is most likely a Web page whose structure and layout is specified in Hypertext Markup Language (HTML). However, the use cases behind HTML did not cover computers interpreting the information in published Web content.

Hence, it is not overstating to say that, initially, the design of the Web was intended to serve human needs. In consequence, even people without background in computer science are part of the Web community when maintaining homepages, sharing thoughts on blogs, or editing open encyclopedias in Wiki systems. Nevertheless, the incredible number of billions of published Web resources may confuse users. Because the WWW as such requires the use of computers for consuming information on the Web, the following question should be raised:

> *Why do computers provide users with so little assistance on consuming information on the Web?*

Berners-Lee et al. [2001] published the idea of the Semantic Web, which extends the WWW by a formal and therefore machine interpretable layer. Here, computers can be told about the information content of a Web resource by describing it formally in the Resource Description Framework (RDF) [Manola et al., 2004]. Web pages of world's largest information providers such as the New York Times, the British Broadcasting Corporation, the White House, the German National Library, and Wikipedia show the great success when adding RDF data as machine-interpretable hints to published information. Actually, by using RDF in Attributes (RDFa) markup it is possible to embed RDF data directly within a Web page's HTML model [Adida et al., 2011].

Figure 1.1 presents the formal value of information on the Web along two axes that represent degrees of underlying syntax and semantics in formalisms. Basic operations on information, such as structured questions or reasoning, require formally represented

information as provided by knowledge representation languages, such as RDF, the RDF Vocabulary Description Language (RDFS), and the Web Ontology Language (OWL).

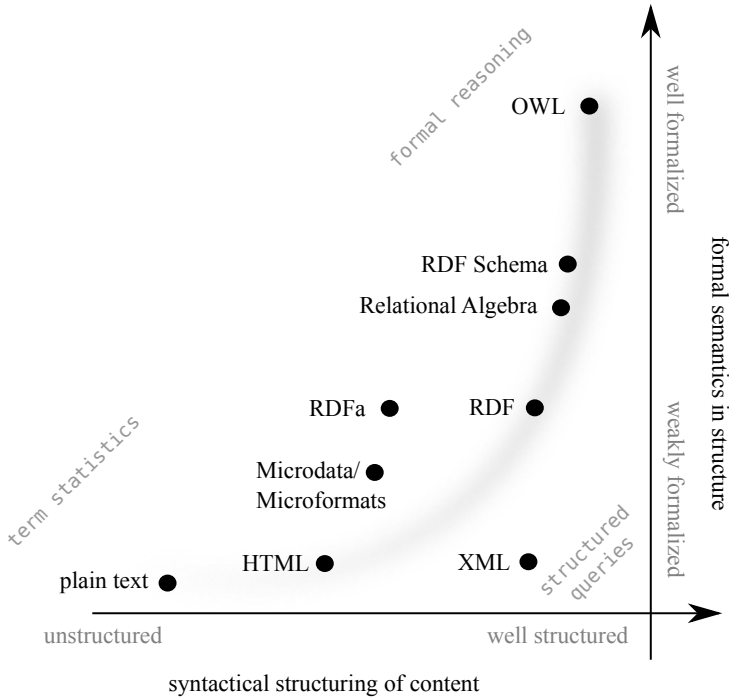


Figure 1.1: Increasing the degree of formalism in specifying syntax and semantics of information results in an increase of the value and the usability of information.

In fact, Web content management systems, such as Drupal or Wordpress support the creation of RDF and RDFa data along publishing processes in Web pages. Nevertheless, most Web pages do not contain any structured information, yet. Still, plain text is of major use in news, education, entertainment, or science. Even those Web pages that do contain RDF data will most likely contain passages of plain text without any machine-interpretable hints. Hence, the following question remains:

> *Who creates RDF(a) data for existing and unstructured Web content and text, respectively?*

1.2 Hypotheses

In Computational Linguistics, the algorithmic process of extracting structured information from unstructured text is referred to as Information Extraction (IE). This work

investigates methods that extend the state-of-the-art in IE by integrating formally represented Semantic Web data into the architecture, the interfaces, and the information extractors of IE systems. This raises the following three research hypotheses:

The utilization of RDF in IE approaches ...

H.1 ... enhances the IE process: Providing Information Extraction (IE)-systems with formal Semantic Web data in RDF enables the investigation of enhanced IE-algorithms that add new opportunities to implementing IE tasks, such as the recognition of entities, the disambiguation of word senses, and the extraction of relations.

H.2 ... facilitates design of IE-interfaces: In IE, templates specify and represent structure and content of information items, which should be extracted from the natural language text. Formal data of the Semantic Web comprises the use of vocabularies to specify the tokens of a certain domain in terms of classes, properties, and relationships of and between individual information items. The incorporation of such vocabularies for defining an information demand on RDF sources in the SPARQL Query Language (SPARQL) facilitates the specification of IE templates. Providing the formal nature of extracted information in RDF facilitates the utilization of extracted results by subsequent applications.

H.3 ... involves domain adaptability: Traditional IE- systems suffer from being specialized on single IE templates and domains of concern. The utilization of RDF-represented information as well as the RDF-represented IE results facilitates the domain independent usage of IE systems. The incorporation of exchangeable formal information involves an increasing flexibility in adapting IE-systems to a new domain of concern.

Each of these research hypotheses are referred to especially from Chapter 5 to 8 covering the main research objectives.

1.3 Contributions

The research activities performed in this work result in the following main contributions. They refer to each of the upper defined research hypotheses.

C.1 Incorporating RDF into IE enables information extractors to utilize additional background knowledge. This work contributes insights on how to pre-process raw RDF data by combining statistical graph and corpus analyzes to compile statistical models that cover single aspects of knowledge patterns. Please refer to Chapters 5 and 6 for more information on pre-processing RDF data.

C.2 Incorporating RDF into IE involves the extension of the traditional Named Entity Recognition (NER). This results in the specification of the Semantic Entity Recognition process, which extends the traditional recognition of named entities with the consumption of formal knowledge provided by RDF graphs. Chapters 5 and 7 provide foundations and detailed information on extracting semantic entities from text.

C.3 By utilizing terms from the formal vocabularies, which are used in RDF data sets, SPARQL queries are applied for specifying IE templates that describe which types of information the IE-system should extract from text. Chapter 8 explains the application of SPARQL to IE.

C.4 In order to deploy the application of RDF-enhanced Information Extraction (IE) into the general architecture of the Semantic Web, this work presents the formalization of IE results in RDF. This also comprises the ability to integrate formal extraction results back into originating Web pages by automatically generating a copy of it enriched with RDF in Attributes (RDFa) markup. Chapter 8 provides detailed information on this topic.

C.5 The state-of-the-art of NER depends on the existence of training data, which is expensive to create. This work proposes a method, which uses the knowledge of an RDF graph to automatically label a text corpus. A labeled corpus is used to train entity classifiers in terms of the domain of concern of the underlying RDF graph. This allows a more general adaption of IE systems to specific application domains. Chapter 6 provides detailed information on the automatic creation of training data.

C.6 Due to the Linking Open Data (LOD) community, a large variety of domains have been published in recent years using RDF data. The evaluations of methods and applications that were developed in this work (see Chapters 6, 7, and 8) were conducted on the data of different RDF graphs. This confirms the general value of an application of RDF to Information Extraction (IE) independently of the underlying domain.

1.4 Prerequisites

The reader should be familiar with general Web technologies including the Hypertext Transfer Protocol (HTTP), the Unique Resource Locator (URL), and its generalized form, the Uniform Resource Identifier (URI). Jacobs and Walsh [2004], Berners-Lee and Fischetti [1999] provide a comprise summary of these technologies.

The author recommends referring to Jurafsky and Martin [2008] for details on Natural Language Processing (NLP), which is applied in Chapter 2 and Chapter 7.

A background in Semantic Web technologies like it is summarized by Hitzler et al. [2009] is helpful to understand its application and utilization in Chapters 3, 5, and 8.

The application of statistics and probabilities in Chapters 5 and 6 requires a basic understanding in these fields of mathematics. Here, knowledge about statistics in machine learning is helpful to understand the computation of models. Duda et al. [2001] and Jurafsky and Martin [2008] provide a comprehensive overview on used machine-learning approaches.

1.5 Terminology

Within this document, references to scientific literature are referring to by keys consisting of the authors' last names and the date of publication. At the end of this document, a literature list provides more citation information about these references. Because some literature is only available online, these references are recorded as standard literature but also consist of a persistent URL. Informative background information related to current topics is referred to by using footnotes.

Within examples and data excerpts of experimental results, RDF graphs are printed in TURTLE syntax Beckett and Berners-Lee [2007]. Here, the following prefixes and namespaces are used to provide a compact illustration:

rdf RDF Vocabulary `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

rdfs RDF Schema Vocabulary `http://www.w3.org/2000/01/rdf-schema#`

owl Web Ontology Language `http://www.w3.org/2002/07/owl#`

foaf Friend-of-a-Friend vocabulary `http://xmlns.com/foaf/0.1/`

dc Dublin core metadata vocabulary `http://purl.org/dc/elements/1.1/`

mo Music Ontology `http://purl.org/ontology/mo/`

dbp DBpedia's resources `http://dbpedia.org/resource/`

dbp-ont DBpedia's ontology `http://dbpedia.org/ontology/`

yago YAGO knowledge base, derived from Wikipedia. `http://dbpedia.org/class/yago/`

geo A vocabulary for representing latitude, longitude and altitude information. `http://www.w3.org/2003/01/geo/`

1.6 Outline

The structure of this document is divided into six parts:

Introduction: Chapter 1 motivates further investigations on IE approaches on the Semantic Web. It exposes three research objectives in Section 1.2 and summarizes the resulting contributions of this research in Section 1.3.

State-of-the-Art / Related Work: The next two chapters outline the state-of-the-art in IE and Semantic Web technologies this work is building upon. Each chapter exposes open issues this work contributes to. More specifically, Chapter 2 outlines the history of recurring problems in IE research from the early beginnings until now. Chapter 3 presents knowledge representation techniques of the Semantic Web for exchanging information in RDF. Finally, the foundations are completed by listing related work on Ontology-based Information Extraction (OBIE) in Chapter 4.

Concept: This part covers the general concept for incorporating RDF for enhancing the implementation of IE tasks. Chapter 5 describes the extension of a named entity with a semantic link associating it with an RDF source. It defines semantic entities and specifies a Semantic Entity Recognition process. Moreover, it outlines formal computation approaches for programming with semantic entities and RDF sources.

Realization: The remainder four chapters describe activities spent on the core research objectives. Chapters 6 and 7 detail on how RDF supports different kinds of information extractors. Each chapter is self-contained in terms of approach, relevant related work, experimental evaluation, and final conclusion. Chapter 6 figures out the preprocessing of textual and formal background knowledge to create models that support the development of concrete information extractors. Within the scope of a general Semantic Entity Recognition process, these tasks are introduced and outlined in Chapter 7. Chapter 8 explains the application of RDF and SPARQL in the input and the output interfaces of an IE system. It describes RDF serialization of extracted information and outlines the use of SPARQL for filtering the information that has to be extracted from text.

Application: Chapter 9 outlines IE software systems and prototypes, which were developed based on RDF-based IE. It describes the implemented RDF-based IE system Service Ontology-based Information Extraction (SCOOBIE), the annotation of Web pages with IE results in Epiphany and finally a Web search extension called Sterntaler.

Conclusion and Outlook: Chapter 10 summarizes, discusses, and concludes the research results and provides hints for promising future work topics.

2 Computation on natural language

Language is the fabric of the Web.

(Hans Uszkoreit, What is Computer Linguistics?, http://www.coli.uni-saarland.de/~hansu/what_is_cl.html, 2000)

A vast amount of information in the Web is represented in unstructured natural language text. In order to extract from text relevant information for users, this work investigates algorithmic approaches processing the natural language in text. In the following chapter, the field of Computational Linguistics is introduced. The necessary foundation of a computational processing of natural language is outlined. Information Extraction (IE) as an application of Computational Linguistics approaches is described in more details. In the remainder chapters, IE techniques will be extended by incorporating formal Semantic Web knowledge.

This chapter introduces the field of Computational Linguistics by focusing on IE tasks that are extended by the research of this work. First, Section 2.1 addresses challenges and visions of Computational Linguistics as a subfield of Artificial Intelligence (AI). Providing a background for the investigations of NLP, Section 2.2 outlines a short historic summary of IE as application of Computational Linguistics and summarizes the state-of-the-art in IE. Subsequently, Section 2.3 summarizes the open challenges in IE. The challenges relate to the research hypotheses claimed in Section 1.2. Finally, Section 2.4 concludes the contributions to these challenges.

2.1 Processing natural language

Computational Linguistics is a research field, which investigates algorithmic NLP methods to finally create a formal understanding of contained information. Hence, from an Artificial Intelligence (AI) point of view, a Computational Linguist's vision is to teach machine agents on how to understand meanings in natural language and on how to apply language in conversations similar to humans.

Jurafsky and Martin [2008] separate the visions of Computational Linguistics into three fields:

- **Conversational agents** that speak and understand natural language like humans.
- The **machine translation** of text from one language into another.
- A **question answering** system that answers questions given in natural language.

The research objectives of this work are related to the question answering. It addresses techniques for extracting structured information from natural language text.

When processing natural language, Jurafsky and Martin [2008] state that, the main challenges occur from resolving ambiguities during its interpretation. For example, the occurrence of *"cook"* in a sentence can be resolved as either verb or noun. In addition, by speaking of *"make"* in a sentence, depending on surrounding context its intended meaning may be *"create"* or *"cook"*. The resolution of language ambiguities is concerned to be a hard problem in AI. It requires formal background knowledge from conversational contexts or about real world facts (please refer to [Shapiro, 1992]).

The application of Computational Linguistics to process Web content, which is in focus of this work, is described by Uszkoreit [2000] as follows[1]:

> *"The rapid growth of the Internet/WWW and the emergence of the information society poses exciting new challenges to language technology. Although the new media combine text, graphics, sound and movies, the whole world of multimedia information can only be structured, indexed and navigated through language. For browsing, navigating, filtering and processing the information on the Web, we need software that can get at the contents of documents.[Uszkoreit, 2000]*

Uszkoreit's description of applying language technology for developing information filtering mechanisms matches exactly with the general intention of this work.

In general, the basic literature about NLP approaches describes a conceptual stack dividing the problem into syntactics, semantics, and pragmatics [Allen, 1994, Jurafsky and Martin, 2008] .

[1] Uszkoreit's definition got cited by the AI topics `http://www.aaai.org/AITopics` of the American Association of Artificial Intelligence (AAAI)

Syntactics: Algorithms within this category address the recognition and processing of well-defined syntactic structures of languages.

Semantics: Approaches performing semantic analysis principally ground on the interpretation of known language semantics.

Pragmatics: Here, the focus is set on the application of knowledge that was interpreted from language.

The following sections outline each of these elements. Later, in Section 2.2.2, syntactics, semantics, and pragmatics are used to classify specific task topics of the state-of-the-art in processing natural language.

2.1.1 Syntactics

Syntactic knowledge about languages specifies basic language segments, such as words and sentences. Syntax rules determine how words can be combined in order to form a correct sentence. They also specify the structural roles of words within the context of a sentence and determine word sequences as phrasal units.

Computational approaches for processing syntax are, for example, grammars specifying syntactic rule sets that allow the generation of valid sentences and parsers for applying these rules when processing language data. Chomsky [1956] described and formalized the computational complexities of natural language grammars and parsers. He showed that even at the bottom of the language processing stack the syntax of natural language may be complex to a form that even the most general model of a computer, i.e., the Turing Machine, is not able to decide whether a sentence is correct according to the formal grammar rules of a language, or not. The following example shows that even the parsing of language segments requires the resolution of language ambiguities.

Example 2.1 (*Syntax*)

Please read the following two English sentences:

"Time flies like an arrow."

"Fruit flies like a banana."

These sentences show that dependent or not the words "Time flies" or "Fruit flies" are determined as noun phrases, an English parser is unable to return decidedly parsing results (see parse tree alternatives in Figure 2.1).

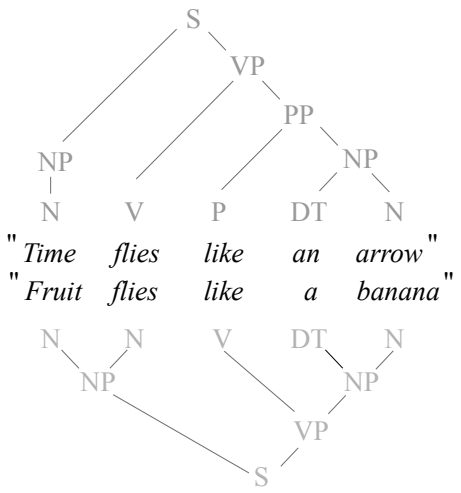


Figure 2.1: Two parse tree alternatives.

2.1.2 Semantics

Language semantics specify the required knowledge used for interpreting the meaning of language segments within the context of a sentence. The disambiguation of word senses, as well as the resolution of co-references created by speaking with pronouns is a typical application of processing language semantics.

Example 2.2 (*Semantics*)
Reading the following beginning of a sentence should not invoke any ambiguities.

> *"We gave the monkeys the bananas,*

But, extending the sentence with the following two options results in ambiguities while resolving the targeting reference of the pronoun "they".

> *... because they were* ***ripe.*** *"*
> *... because they were* ***hungry.*** *"*

Formal language semantics, for example, may add role signatures to phrasal structures. In fact, the roles of monkeys do not comprise monkeys being **ripe** or bananas being **hungry**. This enables a co-reference resolver to decide that the occurrence of they in the first case refers to the bananas, whether the occurrence of they in the second case refers to the monkeys.

2.1.3 Pragmatics

The extraction of formal information is influenced by the interpretation and expectation of a sentence's meaning. Pragmatic approaches comprise the holistic interpretation of a sentence meaning within the context of surrounding sentences as discourse. However, often world knowledge is needed to resolve language ambiguities.

Example 2.3 (*Pragmatics*)
Please read the following text passage carefully:

I was finished and closed the pen ...

Answer yourself the question: What is meant by the speaking of a pen? Keep this answer in mind while continuing reading the next passage to the end of the sentence.

... in order to get the chickens safe this night.

Asking again, about how you would interpret the meaning of pen will most likely generate a different answer. The reason is that now, the interpretation of pen differs because of the extended context, which relates pen with chickens.

2.2 Information Extraction

Jurafsky and Martin [2008] describe IE as a system-oriented application of Computational Linguistics, which is related to the nature of question answering. Grishman [2002] and his team published an overall vision of IE in 2002 on the homepage of project Proteus. Proteus was a driving research project coping with IE.

Definition 2.1 (*Long-term goals of Information Extraction*)
Our long-term goal is

1. *to* ***build systems*** *that*
2. ***automatically find*** *the information*
3. ***you're looking for***,
4. *pick out the* ***most useful bits***,
5. *and* ***present it*** *in your* ***preferred language***,
6. *at the* ***right level of detail***.

Proteus' definition of IE summarizes important high level goals [Grishman, 2002]. The following paragraphs comment single features of this vision and outline the impact each feature has on this work.

1. In general, Grishman's claim to building systems conforms to the overall goal of Computational Linguistics, i.e., to **build systems** that understand natural language. In this work, the system SCOOBIE (see Section 9.1.1) was developed as a prototypical IE system, which verifies the value of utilizing information represented in RDF in the implementations of a number of information extractors.

2. The vision states that information extractors **automatically process** natural language sources and extract contained information items without any supervision. This includes the configuration of parameters used for adapting and applying information extractors to specific RDF data without or at least by requiring only minimal human input. Hence, the goal of this work is to develop IE methods that incorporate and adapt to given RDF data without human supervision. This corresponds with the claimed Hypothesis H.3.

3. *"Extracting the information the user is looking for"* refers to the IE template feature, which enables users to define their information demand from text. In this work, the goal is to enhance the existing IE template mechanisms by incorporating the SPARQL query mechanism. This corresponds with the claim of the Hypothesis H.2.

4. The IE-system should extract or filter only those information items from text the user is currently interested in. This feature entails necessary considerations of **relevance criteria** to rate extracted bits of information. In consequence, this work will use passed RDF data to rate the relevance of the IE-results. This corresponds with the claim of the Hypothesis H.1.

5. The presentation of IE results in a preferred language covers translational aspects of formally representing IE-results. Consequently, IE results should be translated to formats that fit best to the user's need for enabling her to pass IE-results to succeeding applications. This work uses RDF for representing IE results corresponding with the claim of the Hypothesis H.2.

6. IE-systems should not overload the user with verbose details in IE results. Grishman proposes IE-systems to provide summaries about extracted information. The use of Semantic Web technology allows summarizing IE results on a technical level by providing users just URI references that point to more background information about the recognized information items in text. This corresponds with the claim of the Hypothesis H.2 as the applied domain knowledge rates the relevant information in text.

2.2.1 Historic outline

One of the earliest IE systems was developed in the Linguistic String Project. It was based on the application of hand crafted string grammars [Sager, 1967]. The used technology consisted of handcrafted rules and simple template population techniques [Sager, 1981].

DeJong [1979, 1982] proposed the FRUMP IE system. It adapted Schank's high-level script structures for representing episodic knowledge of situations [Schank and Abelson, 1977]. He refined Scripts to what the IE community nowadays refers to as templates, e.g., a model frame with slots that represent bits of knowledge about scenarios such as job changes of top managers. The design of the FRUMP system was intended to populate slots in kind of these template structures.

With the beginning of the Message Understanding Conference (MUC)-series in 1987, a research community emerged and collaborated on developing IE-systems. Results of the MUC-series form the base of all modern IE-systems. During the MUC-decade, independent NLP-tasks, such as Named Entity Recognition, Co-reference Resolution, or Word-sense Disambiguation were defined. A pipeline architecture consisting of a series of transducers emerged as best practice for processing information on different levels and units of language. Finally, the existence of formal IE templates, which define the users' information demand, was a major contribution of the MUC conference series.

Interestingly to see is how the MUC task definitions given to participants changed over time [Grishman and Sundheim, 1996]. Influenced by evolving political and economic interests, the choice of data and extracted content differed from extracting information from naval messages in the late 80-ies, to extract information from Japanese documents about job changes of top managers and joint ventures in the early 90-ies. Finally, the focus was set on the extraction of terror-related information from Arabic texts in the late 90-ies.

The following summary of activities in the MUC-series rests upon "A Brief History of MUC" by Grishman and Sundheim [1996] and surveying work written by Cowie and Lehnert [1996] as well as Wilks [1997].

In the first MUC in 1987, neither a predefined format was given for recording extracted information, nor was a formal evaluation conducted. As result, it was hard to compare the qualities of different systems.

By MUC-2 in 1989, template filling had crystallized as a determining task topic. Developers received a description of a class of events to be identified in the text; for each of the events one had to fill a template with information about the event. IE templates were frame-like structures with slots representing information about a thing which was concerned as an event on these conference call. Ten slots covered information about the event's type, the agent, the time and place, the effect, etc. Precision and Recall measures were used to rate the quality of populated template slots (see Section 7.9.2).

IE templates became more complex comprising 18 slots at MUC-3 in 1991 and 24 slots

at MUC-4 in 1992.

At MUC-5 in 1993, the extraction had to be solved in two languages, English and Japanese. The reason was the increasing influence of Japan in US economy. Here, joint ventures should be extracted from text. This task required 11 templates with a total of 47 slots. The task documentation was over 40 pages long.

```
<TEMPLATE> :=
  DOC_NR  : "NUMBER"
  CONTENT : <SUCCESSION_EVENT> *

<SUCCESSION EVENT> :=
  SUCCESSION_ORG : <ORGANIZATION>
  POST           : "POSITION TITLE"|"no title"
  IN_AND_OUT     : <IN_AND_OUT> +
  VACANCY_REASON : {DEPART_WORKFORCE, REASSIGNMENT, NEW_POST_CREATED, OTH_UNK}

<IN AND OUT> :=
  IO_PERSON     : <PERSON>
  NEW_STATUS    : {IN, IN_ACTING, OUT, OUT_ACTING}
  ON_THE_JOB    : {YES, NO, UNCLEAR}
  OTHER_ORG     : <ORGANIZATION>
  REL_OTHER_ORG : {SAME_ORG, RELATED_ORG, OUTSIDE_ORG}

<ORGANIZATION> :=
  ORG_NAME       : "NAME"
  ORG_ALIAS      : "ALIAS" *
  ORG_DESCRIPTOR : "DESCRIPTOR"
  ORG_TYPE       : {GOVERNMENT, COMPANY, OTHER}
  ORG_LOCALE     : LOCALE-STRING {{LOC_TYPE}} *
  ORG_COUNTRY    : NORMALIZED-COUNTRY-or-REGION | COUNTRY-or-REGION STRING *

<PERSON> :=
  PER_NAME  : "NAME"
  PER_ALIAS : "ALIAS" *
  PER_TITLE : "TITLE" *

LOC_TYPE :: { CITY, PROVINCE, COUNTRY, REGION, UNK }
```

Figure 2.2: Nested IE template in EBNF syntax specifying entities with slots for describing manager successions.

One innovation of MUC-5 was the use of nested template structures. In earlier MUCs, each event had been represented as a single template in effect, a single record in a database, with a large number of attributes. This format proved awkward when an event had several participants (e.g., several victims of a terrorist attack) and one wanted to record a set of facts about each participant. Figure 2.2 contains the Extended Bachus-Naur-Form (EBNF)[2] notation such a template definition.

[2]Please refer to http://www.cs.man.ac.uk/~pjj/bnf/bnf.html for more information on EBNF.

At MUC-5, Hobbs [1993] presented a first technical view on a generic IE architecture:

Definition 2.2 (*Hobb's generic IE architecture*)

An Information Extraction system is a cascade of transducers or modules that at each step add structure and often lose information, hopefully irrelevant, by applying rules that are acquired manually and/or automatically.

[Hobbs, 1993]

One year after MUC-5, Hobbs and Israel [1994] elaborate on knowledge representation techniques for designing templates. The authors were the first to mention ontology in an IE context, i.e., ontology about representing extracted entities.

By MUC-6 in 1995, several goals were defined for specifying the nature of desired information extractors. One goal was to identify task independent component technologies of IE systems.

Named Entity Recognition (NER), for example, was defined as IE task to identify persons, locations, organizations, time expressions, currency, and percentage values in text.

Example 2.4 (*Named entities*)

The example below shows how the Standard Generalized Markup Language (SGML) was used to represent labeled entities in text.

```
The <ENAMEX TYPE="LOCATION">U.K.</ENAMEX> satellite television
broadcaster said its subscriber base grew
<NUMEX TYPE="PERCENT">17.5 percent</NUMEX> during
<TIMEX TYPE="DATE">the past year</TIMEX> to 5.35 million.
```

Furthermore, three "Deep Understanding" IE tasks were proposed:

Co-reference: The IE system has to mark co-referential noun phrases in text passages. Interestingly, Grishman remarked that the initial specification of co-reference analyzes envisioned marking even set-subset; and part-whole relations, in addition to identity relations.

Word-sense Disambiguation: For each open class word (noun, verb, adjective, and adverb) in the text, the IE-system would have to determine its sense (in terms of linguistic semantics). The classification system of the Wordnet thesaurus defining hyponyms and synonyms for English words was used for accomplishing this task [Stark and Riesenfeld, 1998].

Predicate-argument structure By using deep parsing methods, the system would have to create a tree interrelating the constituents of the sentence, using some set of grammatical functional relations.

Up to MUC-6, IE-systems were developed for single domains-of-concerns with specialized extraction tasks. Now, portability was defined as the ability to rapidly re-target a system to extract information about a different class of events. The correct wording of Grishman and Sundheim [1996] speaking about portability in IE was:

> *"The committee felt that it was important to demonstrate that useful extraction systems could be created in a* ***few weeks****."*

The MUC-7 in 1997 was the last conference of the MUC-series. Systems working on NER in English produced near-human performance. The best system scored 93.39% of f-measure. Compared to this, human annotators scored 97.60%.[3]

From 2000–2008 the Automatic Content Extraction (ACE) competition started evaluating IE-systems. The primary ACE research objectives were viewed as the detection and classification of Entities, Relations, and Events in text. In 2009, ACE became a track in the Text Analysis Conference (TAC).

One year later a new IE task called Knowledge Base Population emerged at the TAC 2010. Knowledge Base Population means to use extracted information from text for populating any kind of knowledge base. It was the first IE task being categorized as Ontology-based Information Extraction (OBIE) (see Chapter 4).

2.2.2 State-of-the-Art

Appelt and Israel [1999] provide a tutorial about NLP techniques and IE systems that built on them. Despite the advanced age of this tutorial, it still provides a comprehensive overview about the general architecture most IE systems follow. Jurafsky and Martin [2008] describe IE as the compilation of a series of NLP techniques.

Finite state transducer

A stable architecture pattern implemented in IE systems is to wrap information extractors by a transducer concept such as Finite-state Machines (FSMs) (see Appelt et al. [1993], Kushmerick [2002], McCallum [2002], Bontcheva et al. [2004], Nadeau and Sekine [2007], Adrian and Dengel [2008], Jurafsky and Martin [2008]).

The modeling of FSMs is a well-known technique for processing textual input streams. It is based on states, transitions, and actions. Two groups of FSMs exist, namely recognizers and transducers. Recognizing machines accept certain tokens with a function λ that maps read tokens into the Boolean token set $O := \{false, true\}$. Transducing machines (called finite-state transducer) read tokens and create a new formal language with λ and an output alphabet O as shown in Figure 2.3.

[3] Further readings about MUC-7 results is available under `http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/marsh_slides.pdf`.

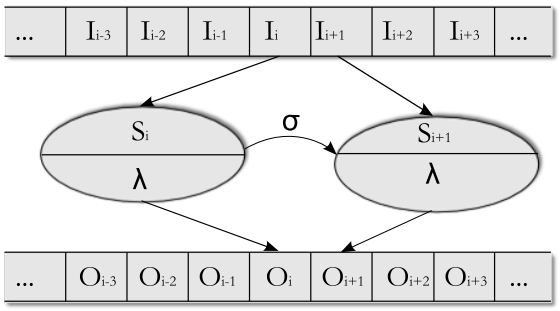


Figure 2.3: A simple extract of a finite-state machine.

According to Chomsky's hierarchy of grammars, a finite-state transducer is capable to implement parsers for regular languages, which is the simplest form of complexity [Chomsky, 1956]. Nevertheless the majority of IE-tasks that are implemented by using this simplistic concept produce sufficient quality. Hence, in this work IE is implemented by means of the system SCOOBIE, built upon this architecture (please refer to Section 9.1.1).

We set the focus on a special type of FSTs called Moore Machines. The basic nature of a Moore Machine is the fact that it generates output based on its current state. Thus, transitions to other states depend on the input.

Definition 2.3 (*Moore Machine*)

We describe a Moore Machine as a 6-tuple $(S, S_0, I, O, \sigma, \lambda)$ *with:*
S *as finite set of states and* $S_0 \in S$ *as initial state,*
I *as input alphabet, and*
O *as output alphabet,*
$\sigma : S \times I \rightarrow S$ *as transition function, and finally*
$\lambda : S \rightarrow O$ *as output function.*

Figure 2.3 outlines a Moore machine between a token input stream I and output stream O. The theory of FST allows composing a series of FSTs to finite-state cascades [Allauzen and Mohri, 2009].

The FSM framework is applied to each of the information extractors in SCOOBIE. The first application of finite-state cascaded within SCOOBIE's pipeline architecture was published in [Adrian and Dengel, 2008]. There, the extension of FSTs with a belief function was proposed adding weights to elements of transducers' input and output alphabets, which are referred to as hypotheses. Within a finite state cascade, this paper describes a combination function with a converging feature that enables transducers to consume multiple weighted hypotheses to finally produce one weighted hypothesis. Later on, the implementation of SCOOBIE's pipeline architecture is restricted to interpret,

weight, and combine only relevance ratings. Calculations on relevance ratings are based on stochastic operations using statistical distributions of these relevance values.

Language identification

Language identification refers to the problem of determining which natural language a text is written in. A common approach is based on the usage of statistics on character n-grams in languages [Dunning, 1994, Cole, 1997]. A typical classifier is then trained on the distribution of a language's character n-grams of length $1 \leq n \leq 5$.

An existing implementation, which provides character n-gram distributions for common languages is used in SCOOBIE (please refer to Section 9.1.1)[4]. Figure 2.4 outlines common tasks of IE systems separated by the general categories syntax, semantic, and pragmatic of Computational Linguistics.

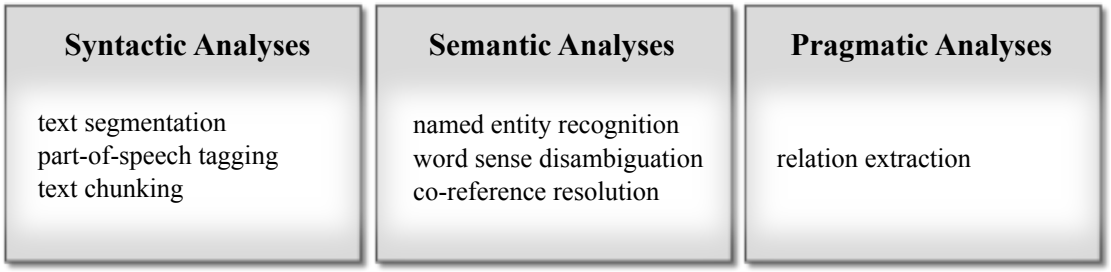


Figure 2.4: General architecture of IE systems.

Text segmentation

Text segmentation describes the problem of segmenting running text into words and sentences [Jurafsky and Martin, 2008]. In European languages, a simple word tokenizer can be implemented by interpreting space characters as delimiters and applying rules that determine the interpretation of existing punctuations for example in abbreviations or sentence boundaries ("e.g.", "i.e."). The segmentation of text into sentences is generally based on the interpretation of punctuations. Certain kinds of punctuations such as "!? :" tend to mark sentence boundaries. Others (e.g., "The 1st Int. Workshop on Text Processing") have to be handled by using abbreviation lists and lexical rules or training classifiers on ground truth data. Finite-state transducers of the Java programming language API are used as word and sentence tokenizers in SCOOBIE (please refer to

[4]The Nutch library provides an open source implementation of a language identifier. See `http://wiki.apache.org/nutch/LanguageIdentifier` for more details.

Section 9.1.1)[5]

Part of speech tagging

In Computational Linguistics, Part-of-speech (POS) tagging, is also referred to as word-category disambiguation. It describes the problem of labeling words in a text to correspond to a particular part of speech. A simplified form of a POS analysis is the identification of words as nouns (NN), verbs (VB), determiners (DT), etc.

Example 2.5 (*POS tagging*)

The following sentence is labeled with part of speech tags:

```
RDF   is    a   graphical knowledge representation format .
NN    VB    DT  JJ        NN        NN             NN     .
```

In English, POS tag sets are defined by the Brown corpus [Greene and Rubin., 1981] or the Penn Treebank[6]. For German, the Tiger corpus provides annotated corpus data Brants et al. [2002].

In this work, POS tagging models for the German and English language provided by the Open NLP library[7] are used. This implementation is based on maximum entropy classifiers [Nigam et al., 1999b]. In Section 7.2 and 7.7, POS tags are used as feature functions that determine a word's meaning in a sentence.

Text chunking

Text chunking describes the problem of classifying words to be part of a phrasal structure. The example below lists a result of a phrase chunking sequence tagger. The labels are specified in the $I - O - B$ notation. B denotes a word to be the beginning of a phrase, I denotes a word to be within a phrasal sequence, and O denotes a word to be outside any phrasal structures. The postfix of $I - O - B$ tags describes the type of phrase. NP denotes a noun phrase. VP denotes a verb phrase.

Example 2.6 (*Text chunking*)

The following sentence is labeled with phrase tags in $I - O - B$ notation.

```
RDF   is    a   graphical knowledge representation format .
B-NP  B-VP  O   B-NP      I-NP      I-NP           I-NP   O
```

For implementing a text chunker, the shared task on text chunking of the Conference on Computational Natural Language Learning (ConLL) in 2000 provides a data set[8]

[5] http://download.oracle.com/javase/6/docs/api/java/text/BreakIterator.html
[6] The data set is available at: http://www.cis.upenn.edu/~treebank/
[7] http://incubator.apache.org/opennlp/
[8] The data set is available at: http://www.clips.ua.ac.be/conll2000/chunking/

annotated with phrase labels. Tjong Kim Sang and Buchholz [2000] present a general overview of the systems that have taken part in the shared task and briefly discuss their performance.

One goal in IE is to detect names in text that refer to real world concepts. In this context, a further investigation on the use of nouns in text is needed: Nouns are traditionally grouped into proper nouns and common nouns. **Proper nouns**, like Regina, Colorado, and IBM, are names of specific persons or entities. They are also referred to as proper names. **Common nouns** are divided into count nouns and mass nouns. **Count nouns** are those that allow grammatical enumeration; that is, they can occur in both the singular and the plural (goat/goats, relationship/relationships) and they can be counted (one goat, two goats). **Mass nouns** are used when something is conceptualized as homogeneous group. Therefore, words like snow, salt, and communism are not counted [Jurafsky and Martin, 2008].

In this work, text chunking, for example, is applied to identify noun phrases in text (refer to Section 7.2). These syntactic language segments are considered to be candidates for proper names and therefore provide a good starting point for a subsequent semantic analysis such as the recognition of named entities.

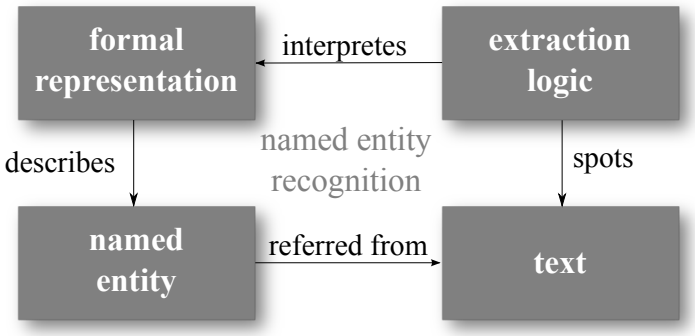


Figure 2.5: Components used for recognizing entities in text.

Named Entity Recognition

In contrast to just formally analyzing the syntax of proper names, NER refers to the problem of labeling word sequences as proper names and classifying the type of entity they refer to. The MUC series defined a fixed set of types IE-system should recognize in text mainly consisting of person, location, organization, and event.

The general architecture for building a named entity recognizer is given by Figure 2.5. First, a machine understandable representation for describing the types of entities is needed. Such representation often depends on large lists of known names and values of

these types. In literature, these lists are often referred to as dictionary or gazetteer.

Rule-based approaches model finite-state automaton on these lists for spotting their entries in text. For example, the GATE framework provides such an implementation [Cunningham et al., 2002]. GATE also provides the JAPE language [Cunningham et al., 2000] for specifying lexical rules that enable domain experts to create entity representation by combining values of gazetteers. JAPE also allows the specification of regular expressions for a more generalized definition on possible entity values.

The state-of-the-art in NER applies machine-learning models to train sequence taggers. In general, these supervised machine-learning techniques require a corpus of examples to exist. These examples are labeled with entity classes that should be learned and finally predicted by the sequence tagger. Sequence taggers analyze statistical distributions on word co-occurrences for each label in the corpus. These distributions create discriminating evidences for labeling a word sequence correctly. Freitag [1998, 2000] proposed the application of such supervised machine learning techniques to NER and IE in general. Commonly used machine-learning models for NER are Hidden Markov Models [Todorovic et al., 2008] or Conditional Random Fields (CRFs) [Finkel et al., 2005] (see Section 5.4.8).

Existing corpora are provided by the shared task on language-independent NER at ConLL 2003[9] challenge. Tjong Kim Sang and De Meulder [2003] presented a general overview of the systems that have taken part in the shared task and briefly discuss their performance.

With the series of Pascal challenges, Ireson et al. [2005] organized a challenge on evaluating machine-learning approaches for IE and therefore provided a corpus. In this work both corpora are used to train sequence taggers and rate the quality of their labeling results.

Indicator functions form the elementary features for describing the representation of entities in statistical distributions. These functions are binary functions returning one if for example a certain word exists within the observed context of a possible entity candidate. In general, such features can be categorized as describing either the context or the content of an entity. Figure 2.6 lists different implementations of both categories.

A popular approach for representing entities content-based is to use plain lists of possible values. In their survey on NER, Nadeau and Sekine [2007] describe the usage of name lists as privileged. They conclude that most supervised machine-learning methods read large annotated corpora and memorize these large lists of entities. They also state that most list-based approaches require candidate words to match at least one element of a pre-existing list.

[9] The data set is available at: `http://www.cnts.ua.ac.be/conll2003/ner`

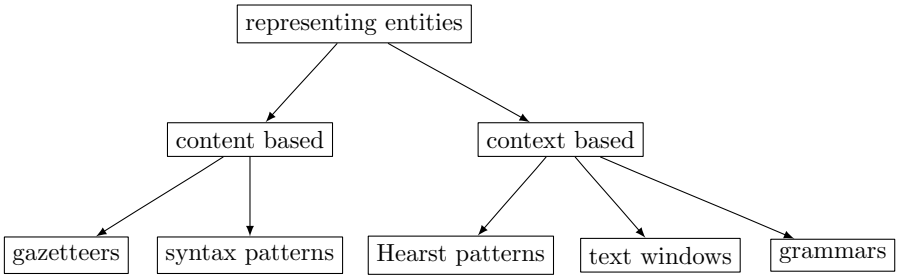


Figure 2.6: Feature types used for representing named entities.

Example 2.7 (*Name lists*)

The following word lists contain different kind of entity names.

```
person first names    = Horst, Harald, Konrad
person family names   = Zuse, Finke, Merkel
company suffixes      = GmbH, Inc., AG, Holding
```

The disadvantage of the list-based approach is that it is hardly possible to collect all possible values of an entity type within a list. In consequence, these lists turn out to be very large and in terms of family names carry millions of entries. Because of polysemy, the same word often occurs in more than a single list (for example, my family name Adrian may likely occur in a list of first names. Mercedes may name a girl or a car.). Hence, disambiguation rules have to be created for these names.

In cases where entity values possess an underlying syntax structure, the disadvantage of gazetteers can be overcome by formalizing this structure with syntax patterns. Entities of this category are also referred to as structured entities. A simple formal description of a structured entity like a date can be defined in a regular expression.

Example 2.8 (*Regular expression*)

The following regular expression defines a simple shape of a date expression.

```
([1-9][0-9]^{3})-([0-1][0-9])-([0-3][0-9])
```

It is written in Perl syntax and describes a date as three groups of digits separated by a dash. The first group represents the year by four digits and is restricted to hold values in a range from 1000 *to* 9999*. The second group represents a month by two digits. In order to keep it simple, the representation of months is a bit underspecified, as it allows values in a range from* 01 *to* 19*. The last group represents a day by two digits in a range from* 00 *to* 39*, which is again a bit underspecified.*

More general syntactic patterns descriptions regarding cases, morphologies, punctuations, and digits within entity values are given in [Nadeau and Sekine, 2007].

Apart from describing the lexical content of an entity, it is possible to describe information about the linguistic context surrounding an entity in text. The following example gives an impression about handling linguistic context.

Example 2.9 (*Linguistic context*)

The following sentence contains a role relation between multiple entity references.

***Situated in** Silicon Valley, Mountain View **is home to many high technology companies**.*

The named entity of type city "Mountain View" occurs nearby another named entity "Silicon Valley" of type location.

The role signatures of "is home to many high technology companies" and "situated in" classify "Mountain View" as location.

Common descriptions of a linguistic context are either based on grammars and parsers consisting of hand-crafted language rules [Nadeau and Sekine, 2007], on statistical or probabilistic distributions on word n-grams in sliding text windows [Todorovic et al., 2008], or they are based on lexical rules (also described as local text grammars) called Hearst patterns [Hearst, 1992].

Rule-based approaches formalize signatures of verb, adjective and adverb usages in grammars for inferring the type of a named entity. Disadvantages are the high costs for creating and maintaining these rules. Rules are hardly adaptable to other domains of concerns or languages. Machine-learning-based approaches built models that learn dependencies between co-occurring entity types and surrounding words from annotated text corpora. Hearst patterns are a method for the automatic acquisition of the hyponymy lexical relation from text. They are based on simple lexico-syntactic patterns as shown in the following example:

Example 2.10 (*Hearst pattern*)

This Hearst pattern will extract two facts from this text snippet:

> *"a machine learning model, such as Maximum Entropy or Conditional Random Fields"*

1. *hyponym (Maximum Entropy, machine learning model)*
2. *hyponym (Conditional Random Field, machine learning model)*

$$NP_0, \textit{ such as } NP_1, NP_2, \ldots, NP_{n-1}(\textit{ and}|\textit{or }) NP_n$$
$$\textit{for all } NP_i, 1 < i < n, hyponym(NP_i, NP_0)$$

Cimiano et al. [2004] applied Hearst patterns to populate a formal knowledge base. They faced the fact that such patterns occur rarely in text. NER approaches are used and refined within the Semantic Entity Recognition process in Section 7.4.

Word-Sense Disambiguation

Ambiguities in the semantic interpretation of language are already discussed in Section 2.1.2. It cannot be ensured that the semantics of each language unit, such as a name, is that clear that the intended entity reference can be resolved correctly in all cases. Navigli [2009] describe the technology of Word-sense Disambiguation to determine, which sense of a word is activated by its use in a particular context. If a word conveys only a single meaning, it can be described as **monosemous**. Conversely, **polysemous** words are words conveying more than one meaning. If the different meanings of a single word do not share any relations, the word is referred to as **homonymous**.

Disambiguation methods that utilize external knowledge sources, i.e., RDF data, are addressed in Section 7.5. Knowledge sources are valuable for Word-sense Disambiguation if they provide data, which formalize associations between words and senses. Apart from approaches that use dictionaries or linguistic thesauri such as WordNet, Navigli [2009] outlines the application of formal ontologies consisting of formal taxonomies and general semantic relations as valuable knowledge source.

Co-reference resolution

Two noun phrases co-refer to each other if both of them resolve to the same unique referent [Elango, 2005]. A special kind of co-reference is denoted by the Anaphora:

Definition 2.4 (*Anaphora*)

If A and B are noun phrases and if the interpretation of B requires the interpretation of A the relation between B and A is called anaphora.

Example 2.11 (*Co-reference vs. Anaphora*)

The following two sentences describe a co-reference and an anaphoric relationship.

Co-reference *Peter Parker shot photos of Spiderman.*

Anaphora *Peter Parker put on his uniform.*

The shared task of ConLL 2011 was concerned about a co-reference resolution.[10] Within the scope of this work, anaphoric relations were processed in terms of pronouns. Due to the existence of detailed instance descriptions in Semantic Web data concerning multiple names (please refer to Section 5.2 on RDF components for detailed information on modeling names), the resolution of co-references is obsolete for recognizing entity references in text as described in Section 7.4.

A simple resolution of pronouns was performed for classifying noun phrases in text (see Section 7.7). Here, feature representations of noun phrases consist of the information with the sentence each noun phrase occurs in. The feature representation is extended with information from subsequent sentences if these start with a pronoun, e.g., "Angela Merkel is born in Hamburg. She is the German chancellor".

Relation extraction

Co-occurring entities in a sentence may induce the existence of an explicit relationship in between them. Relation extraction techniques try to recognize such relationships between two or more entities in text.

Example 2.12 (*Relation extraction*)

*"The headquarters of **Google** are situated in **Mountain View**."*

In this sentence, "Google" and "Mountain View" are given as named entities. The goal of a relation extractor is to recognize the relation between both entities as an organization to location associations, which may be labeled like `situated_in`.

Describing the state-of-the-art in relation extraction, Bach and Badaskar [2007] and Jurafsky and Martin [2008] categorize existing approaches in supervised and unsuper-

[10] More details on the task description, training data, and final system results can be retrieved at `http://conll.bbn.com/`.

vised methods.

Supervised methods Bach and Badaskar [2007] describe supervised relation extraction approaches as classifiers. For a given set of relations, the classifier predicts the best matching relation for a passed text segment. Due to the supervised nature, labeled corpus data is needed for training the classifier on example text segments. The challenge in this field is determined by finding a suitable feature representation of relations based on the given text segments. Common approaches are based on simple text-based features such as bags of word n-grams or more complex features based on parse trees. Classifier models such as support vector machines, maximum entropy models, or CRFs are trained upon these features. Concluding the supervised approach, Bach and Badaskar [2007] reveal the following limitations:

1. For each type of relationship, labeled training data is needed.
2. Recognition of n-ary relationships between more than two entities is difficult.
3. The use of text-based features is computationally burdensome and does not scale with increasing amounts of data.

Semi-supervised methods Hearst [1992] propose an approach describing taxonomic relationships between entities by using local text grammars (see Section 2.2.2). The approach starts with a small set of lexical patterns, which are referred to as seed. Iteratively matches of these rules in text are processed. If two known entities co-occur within similar lexical textual contexts, a new lexical pattern is created and added to the seed. Depending on the initial seed and the used text corpus, this approach converges or diverges by creating new rules within each iteration. Cimiano et al. [2004] adapt this approach to recognize semantic relationships but revealed that the recall of recognized relationships remains on low level. However, Bach and Badaskar [2007] and Jurafsky and Martin [2008] report that the application of semi-supervised methods to recognize semantic relations have not been changing much compared to the originating Hearst patterns.

Section 7.8 describes methods, which interpret co-occurring named entities by considering existing relationships between these entities within the given RDF data.

2.3 Unsolved and emerging challenges

Since the MUC decade, the methodology of IE has not been extended, significantly yet. Until now, the community has raised a list of open issues on IE that relate to the research hypotheses in Section 1.2:

Formalize IE-interfaces: During the MUC conference series the constant increase of complexity of IE templates resulted in a high complexity of template specifications [Gaizauskas and Wilks, 1998]. By now, IE templates are created and described manually by domain experts. A single IE template description could extend forty pages in MUC-5.[11] Wilks [1997] criticizes the limits of IE templates as being far too over-specified and too technical for being used in real world applications. Therefore, he demanded a practical and cost effective construction and adaptation of templates to new domains.

Hobbs and Israel [1994] stated very early that templates are devices for ontological knowledge representation. However, a formal template representation like an IE-template definition language is missing in this field.

Provide IE with domain adaptability: Yangarber and Grishman [1997] state that the customization of IE-systems is crucial, as it directly affects every aspect of the progress in the field. Nevertheless, the schedule of MUC-7 did not comprise any tasks concerning customization and domain adaptability. By now, no IE-system exists that solves this issue in portability. Wilks [1997] and Gaizauskas and Wilks [1998] complain that porting IE-systems to new domains is a serious bottleneck for state-of-the-art systems. With respect to adaptation costs, Cowie and Lehnert [1996] state the following:

> *The actual cost of building an IE system is unknown, but estimates at a recent Tipster workshop suggest it should take from* ***three to six person-months*** *for a computational linguist to port a system to a new domain [Cowie and Lehnert, 1996].*

Currently, this estimation of **three to six person-months** is far from being applicable for business or personal use cases. Taking the judgmental statements about traditional IE approaches Grishman [1997], Cowie and Wilks [2000], and Cunningham [2006] conclude three main challenges:

i. Facilitate maintainability for reacting on evolving domain knowledge in IE system, on demand.

ii. Increase re-usability for refocusing existing IE systems to other domains, rapidly.

iii. Enhance usability of templates in order to allow ad-hoc template generation, facilitate template filling, and reduce the complexity of template unification.

Concluding the survey and template shortcomings, a forth challenge was added:

iv. Separate background knowledge specifications from templates and use background ontologies for knowledge representation purpose.

[11] MUC-6 template definitions can be reviewed under `http://www.aclweb.org/anthology-new/M/M95/M95-1027.pdf`.

2.4 Contributions

The research hypotheses (see Section 1.2) address exactly these major shortcomings of traditional IE-systems:

- With respect to Hypothesis H.1, the incorporation RDF data enables the exchange and utilization of domain knowledge. The approaches presented in Chapter 5 cover these basic aspects of letting information extractors compute with knowledge in form of RDF.
- On the basis of Hypothesis H.2, this work investigates the use of SPARQL for specifying IE templates in terms of formal queries. This enables users to specify an information demand by re-using the vocabulary of the existing domain knowledge. Hence, knowledge engineering is no longer required for specifying IE templates. The knowledge is already defined in the schema behind the given RDF data. Chapter 8 illustrates the concrete use and value of SPARQL in RDF-based IE systems.
- The contribution of Hypothesis H.3 states the automatic adaptation of an IE-system to the patterns of formal domain descriptions in RDF. The complete design of the RDF-based IE process in Chapters 5, 6, and 7 is driven by the requirement to adapt systems from one RDF source to another.

3 Representing knowledge on the Web

A new form of Web content that is meaningful to computers . . .

(Berners-Lee et al. [2001])

Information that is represented in natural language text is difficult to interpret by computers. By nature, the WWW provides a large amount of natural language sources in Web pages. Consequently, the Semantic Web offers formal knowledge representation techniques for translating unstructured information meaningful to computers. This work investigates methods, which incorporate information, formally represented in RDF, into information extractors approaches. The goal is to utilize the bits of formal domain descriptions to facilitate the adaption to these domains of concerns and to enhance performance of information extractors.

This chapter provides required background on the Semantic Web technology. Section 3.1 outlines the general vision of the Semantic Web. Next, in Section 3.2 RDF is elucidated for exchanging and representing information. In Section 3.3, techniques are presented for aggregating information about individual instances. The use of formal properties and class hierarchies is addressed in Sections 3.4 and 3.5. Section 3.6 describes the notion of naming graphs with URIs. Vocabularies and ontologies are introduced in Section 3.7 for translating knowledge formally on the Web. SPARQL is presented in Section 3.8 for describing and querying specific parts of RDF data. In Section 3.9, Linking Open Data (LOD) as form of publishing RDF data is introduced for being utilized by IE systems. Section 3.10 concludes related issues of the Semantic Web to the research hypotheses. Finally, Section 3.11 summarizes the contributions of this work to the vision of the Semantic Web.

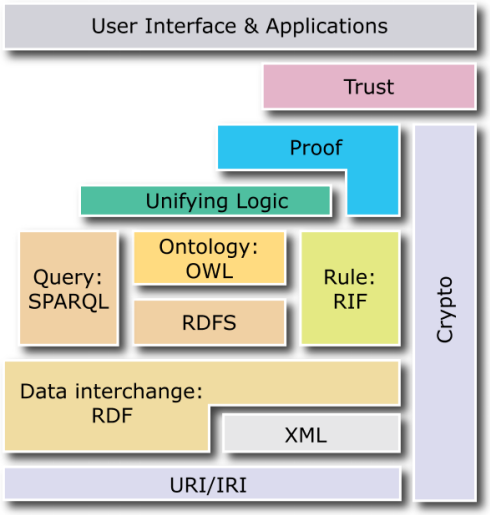


Figure 3.1: The Semantic Web layer cake (W3C, 2007).

3.1 Computing with semantics in the Semantic Web

In 2001, Tim Berners-Lee (inventor of the WWW), James Hendler and Ora Lassila described the vision of a Semantic Web as follows [Berners-Lee et al., 2001]:

> *"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation".*

Here, by speaking of "the current one", the authors refer to the WWW consisting of:

- Web documents written in Hypertext Markup Language (HTML) or Extensible Hypertext Markup Language (XHTML),
- Web resources identified by a Uniform Resource Identifier (URI),
- hyperlinks between Web resources,
- and finally the Hypertext Transfer Protocol (HTTP).

The Semantic Web layer cake depicted in Figure 3.1[1] is an official attempt to summarize the integration of Semantic Web technology. URIs and its extension Internationalized Resource Identifier (IRI) form the basis of addressing and identifying resources

[1]This official version of the Semantic Web Layer Cake is copyrighted by W3C and published at this persistent URL: `http://www.w3.org/2007/03/layerCake.png`.

on the Web. The Extensible Markup Language (XML) enables the markup of plain text with additional structures. The design of XML is document-oriented. Therefore, its underlying mental model is a tree hierarchy. It is used to markup the structure of Web documents with languages such as (X)HTML. For exchanging information about Web resources, the Resource Description Framework (RDF) is applied. In contrast to XML, RDF does not imply a hierarchical but a graph model. The reason is that whereas XML is used to formalize document structures, RDF formalizes link structures of and between resources. Here, symmetries and inverse relation require the existence of circular relationships. In order to give RDF data a well-defined meaning, vocabularies, rules, and formal ontologies can be created by using the RDF Vocabulary Description Language (RDFS), which is also referred to as RDF Schema, the Rule Interchange Format (RIF), and the Web Ontology Language (OWL). The SPARQL Query Language (SPARQL) allows querying RDF data with formal graph queries. Above these data models, the unifying logic intends to mashup different RDF graphs without losing semantics. What is called "proof" intends to check data correctness and data integrity in order to guarantee data quality. The "trust" layer allows users to select only data from originators of their confidence. Finally, "crypto" is a means that adds security to a Web of sensitive data.

Extending the Semantic Web layer cake, results of this work enable the automatic translation of information from natural language text into formally represented information in RDF.

3.2 Resource Description Framework

RDF is the standard for exchanging formal information on the Web. Its design encompasses the separation of schema and raw data. This enables the merging of information from different sources even if the underlying schemes differ [Manola et al., 2004].

For transferring RDF statements on the Web, RDF provides a couple of serialization formats. Some of them are specified in Extensible Markup Language (XML) in order to simplify the validation and parsing of RDF data (i.e., RDF/XML[Becket, 2004] or TRIX[Carroll and Stickler, 2004]). The nature of XML is verbose and its tree model invokes problems when serializing graphs. Therefore, RDF can also be serialized in plain text or line based formats (i.e., Notation 3 [Berners-Lee and Connolly, 2008] or its sub-language Turtle [Beckett and Berners-Lee, 2007]).

"RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link."[Manola et al., 2004] An RDF statement consisting of two resources and a link in between is usually referred to as a "triple". *"The linking structure of RDF forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual*

```
@prefix foaf:       <http://xmlns.com/foaf/0.1/>   .
@prefix dbp:        <http://dbpedia.org/resource/> .
@prefix fb:         <http://rdf.freebase.com/ns/fb:time.event.> .
@prefix dbp_ont:    <http://dbpedia.org/ontology/> .
@prefix xsd:        <http://www.w3.org/2001/XMLSchema#> .

dbp:Battle_of_Kaiserslautern
    foaf:name               "Battle of Kaiserslautern"@en ;
    fb:start_date           "1793-11-28"^^xsd:date ;
    fb:end_date             "1793-11-30"^^xsd:date ;
    dbp_ont:place           dbp:Kaiserslautern .

dbp:Kaiserslautern
    foaf:name               "Kaiserslautern""@en .
```

Figure 3.2: RDF graph in TURTLE syntax. It starts with prefix definitions of vocabulary shortcuts creating aliases for namespaces. Dots denote the end of a statement. Semicolons denote that succeeding statements are about the same subject. The graph describes the "Battle of Kaiserslautern".

explanations." [Manola et al., 2004]

Representing knowledge in RDF enables machines to interpret it as a list of triples. The example in Figure 3.2 contains RDF statements that give information about the "Battle of Kaiserslautern". It shows triples consisting of a *subject* of which this statement is about, a *predicate* that identifies the type of link (called property), and finally an *object* that is the property value. The same knowledge is rendered as graph in Figure 3.3.

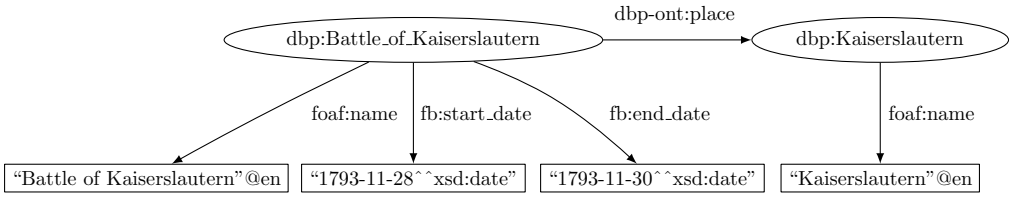


Figure 3.3: Graph visualization of RDF data about the "Battle of Kaiserslautern".

RDF defines the following data types to be used within RDF triples:

URI references are tokens assigning a formal name to any kind of referent. Hence, URI references represent the existence of entities and are used to describe these with additional properties. Subjects of RDF triples are assigned with a URI reference to determine the general subject described by this statement. Assigning a URI reference to a triple's predicate determines the relationship between subject and

object. In order to describe relationships, URI references used within predicates may also be used as subjects in triples [see Klyne and Carroll, 2004].

Blank Nodes are treated as simply indicating the existence of an entity, without using or saying anything about the formal name of that thing [see Hayes, 2004].

Literals are used to identify values, such as numbers and dates, by means of a lexical representation [see Klyne and Carroll, 2004]. RDF distinguishes between **plain literals** that have an optional language tag (e.g., `"Battle of Kaiserslautern"@en`) and **typed literals** which refer to the datatype of the literal's lexical value by a URI (e.g., `"1793-11-28"^^xsd:Date`).

In the remainder of this document, URI references and blank nodes, which occur as subject in RDF statements inside RDF graphs are referred to as **instances**.

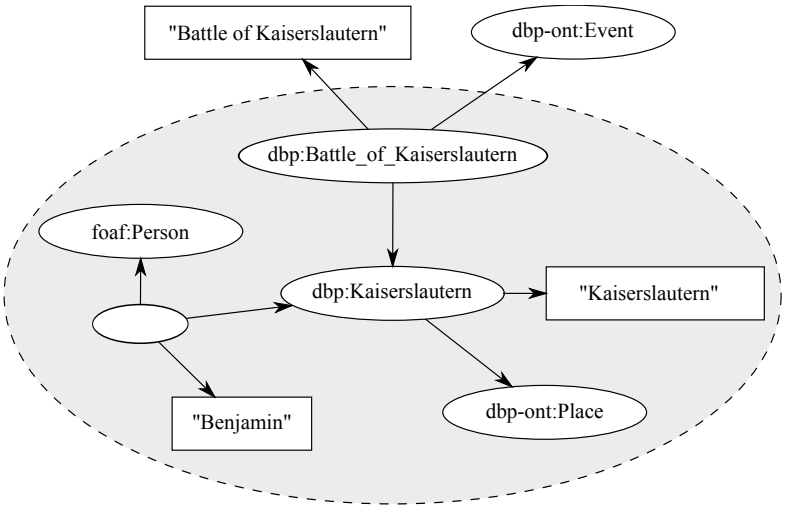


Figure 3.4: Symmetric Concise Bound Description of `dbp:Kaiserslautern`.

3.3 Representing instance knowledge

In general, an RDF graph provides information about more than a single instance. The symmetric Concise Bound Description (CBD) [Stickler, 2005] is a method for separating an RDF graph into logical sub graphs, each consisting of information about a single instance. The algorithm to extract a symmetric CBD about a specific subject from an RDF graph is defined as follows [Stickler, 2005]:

Algorithm 3.1 (*Symmetric Concise Bound Description (CBD)*)

Input: *Given (i) a* ***starting node*** *in an RDF graph and (ii) a sub graph of that* ***source graph****.*

Output: *It returns a symmetric concise bounded description of the instance denoted by the starting node.*

1. *Include in the sub graph all statements in the source graph where the subject of the statement is the starting node;*
2. *Recursively, for all statements identified in the sub graph having a blank node object:*
   → *include in the sub graph all statements in the source graph where the subject of the statement is the blank node in question and which are not already included in the sub graph;*
3. *Include in the sub graph all statements in the source graph where the object of the statement is the starting node;*
4. *Recursively, for all statements identified in the sub graph having a blank node subject not equal to the starting node:*
   → *include in the sub graph all statements in the source graph where the object of the statement is the blank node in question and which are not already included in the sub graph.*

By using the symmetric CBD, it is possible to collect existing information about an instance into a single sub graph. Figure 3.4 illustrates an example RDF graph and highlights the CBD of the instance `dbp:Kaiserslautern`.

3.4 Representing properties

Within an RDF graph, RDF statements consist of different types of property references. Some statements assign literal values to subjects others connect two URI references. The value of each kind of these statements differs when consumed by information extractors. Hence, these properties were separated into three categories:

RDF allows defining slot-like attributes as *datatype properties*.

Definition 3.1 (*Data properties*)
RDF statements assign literal values to subjects. The OWL 2 vocabulary defines RDF properties with a defined range of literals data properties [Bock et al., 2009]. e.g.,

```
dbp:Konrad_Zuse    foaf:name    ''Konrad Zuse''    .
```

In RDF, classifications are represented by using the *type property*.

Definition 3.2 (*Type property*)
RDF statements subsume subjects under defined classes. For this reason, the RDF vocabulary provides an explicit RDF type property [Klyne and Carroll, 2004]. e.g.,

```
dbp:Konrad_Zuse    rdf:type    foaf:Person    .
```

In RDF relations between two instances are formalized by using *object properties*.

Definition 3.3 (*Object properties*)
RDF statements connect two subjects by using an RDF property. The OWL 2 vocabulary defines RDF properties with a defined range of resources that are not literals object properties [Bock et al., 2009]. e.g.,

```
dbp:Konrad_Zuse    dbp-owl:known_for    dbp:Z4_(computer)    .
```

3.5 Representing classes

In RDF, the description of real world entities comprises the classification of those along one or multiple hierarchies of RDFS classes. A class in RDFS summarizes subjects that share a similar nature. In some cases this shared nature is explicated by a set of shared properties whose domain or range is restricted on subjects of this class.

More technically, the RDF Primer [Manola et al., 2004] defines the meaning of classes as follows:

> *"Classes can be used to represent almost any category of thing, such as Web pages, people, document types, databases, or abstract concepts. Classes are described using the RDF Schema resources* `rdfs:Class` *and* `rdfs:Resource`*, and the properties* `rdf:type` *and* `rdfs:subClassOf`*."*

In RDF-based IE, classes will be used for describing the nature of recognized entities (see Section 5.2).

3.6 Representing multiple RDF graphs

Carroll et al. [2005] proposed the concept of having multiple RDF graphs in a single document, and naming them with URIs. Therefore, the TRIX syntax was invented for serializing RDF triples in XML syntax and grouping them in terms of graphs [Carroll and Stickler, 2004]. Bizer and Cyganiak [2007] extended the TURTLE syntax by defining the TRIG syntax, which was able to express graph URIs as well. The following example illustrates an RDF document containing two named graphs:

Example 3.1 (*Serializing named graphs in TRIG syntax*)

```
<http://example.org> = {
 <http://example.org> dc:title ''Queen and Paul Rogers'' .
} .

<http://dbpedia.org> = {
 dbp:Brian_May          mo:member dbp:Queen .
 dbp:Roger_Taylor       mo:member dbp:Queen .
 dbp:Freddy_Mercury     mo:member dbp:Queen .
} .
```

Later on named graphs will be used to serialize IE results in RDF (see Section 8.3).

3.7 Vocabularies

Representing knowledge in RDF is independent from existing schemes. In consequence, just processing RDF triples with machines does not provide any facilities of a formal understanding that is required to proof statements or infer new ones. Therefore, vocabularies are used to describe a schema that underlies the RDF data.

"On the Semantic Web, vocabularies define terms like concepts and relationships to describe and represent an area of concern. Vocabularies are used to formalize the terms that can be used in a particular application"[W3C, 2010b]. For example, Friend of a Friend (FOAF) is a vocabulary devoted to linking people and information using the Web [Brickley and Miller, 2010]. *"Vocabularies characterize possible relationships, and define possible constraints"*[W3C, 2010b]. For example in FOAF, the relationship knows is represented by the token `foaf:knows`. It links two resources that are described as being persons in terms of `foaf:Person`.

Examples of other vocabularies are RDFS, and the DBpedia Ontology. RDFS is a vocabulary for using RDF to describe RDF vocabularies [Brickley and Guha, 2004]. The DBpedia Ontology is a shallow, cross-domain vocabulary based on the most commonly used infoboxes within Wikipedia [Bizer et al., 2009b].

There is no clear division between what is referred to as "vocabularies" and "ontologies" on the Web. In fact, the common use of the term ontology tends to be over

generalized. Gruber [1993] provided a high-level definition of ontology as being a shared conceptualization, which classifies each vocabulary in use by the Semantic Web community as ontology. *The trend is to use the word "ontology" for more complex, and possibly quite formal collection of terms, whereas "vocabulary" is used when such strict formalism is not necessarily used or only in a very loose sense* [W3C, 2010b]. This means that, for example, the Dublin Core vocabulary, which describes terms for adding general metadata to Web resources, may be considered as plain vocabulary. Data in Dublin Core do not facilitate any formal inference. Nevertheless, Dublin Core is the vocabulary that is most frequently used. In contrast, the Creative Commons vocabulary specifies terms that formalize licensing restrictions on re-using published Web resources. As the existence of Creative Commons data about resources has severe implications on the application and further processing of such resources, it might be referred to as ontology. *However, vocabularies are the basic building blocks for inference techniques on the Semantic Web* [W3C, 2010b]. Vocabularies provide a portable representation of machine-interpretable knowledge on the Web. In this work, the term vocabulary is preferred, as formal reasoning beyond simple entailment is not in focus of IE on the Semantic Web, yet. The use of vocabularies as underlying schemes can be seen in Example 3.2.

Example 3.2 (*Classification*)

An instance is classified as `db_ont:Town` *and* `dbp_ont:Location`*.*

```
@prefix dbp:      <http://dbpedia.org/resource/>                .
@prefix dbp_ont:  <http://dbpedia.org/ontology/>                .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#>       .

dbp:Kaiserslautern  rdf:type        dbpedia_ont:Town     .
dbp_ont:Town        rdfs:subClassOf dbpedia_ont:Location .
```

The subclass relationship defines a subsumption between Town and Location. This assertion let a reasoner infer the following RDF statement:

```
dbp:Kaiserslautern rdf:type dbp_ont:Location .
```

If a computer is given a schema with defined rules, it is enabled to validate and entail a given data set that follows this schema. Still, the machine does not "know" what a town or location is, but it knows that every town is a location. This circumstance (called "Chinese Room Argument" by John Searle in [Searle, 1980]) should give a realistic impression on how to read the notion machine-understandable.

3.8 Querying with SPARQL

"SPARQL is the query language for RDF data. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions.

The results of SPARQL queries can be result sets or RDF graphs"[SPARQL Working Group, 2008]. By using SPARQL, it is possible to query an RDF graph for existing sub graphs. Thereby, the SPARQL language allows different query types [Prud'hommeaux and Seaborne, 2008] that return different result formats:

SELECT Returns the variables bound in a query pattern match. Figure 3.1 shows such a select query.

CONSTRUCT Returns an RDF graph constructed by substituting variables in a set of triple templates.

"With SPARQL queries on RDF data can be specified across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware" [SPARQL Working Group, 2008]. In fact, the approach described in this work is exactly such a middleware. One claim of this work is devoted to integrate the SPARQL Query Language (SPARQL) into IE systems in order to specify and filter what information the system must extract from text. Using the Query 3.1 as such, a filter would return only those entities from text that are German women in Politics having the first name "Angela". The use of such SPARQL queries is expected to be much more usable than specifying traditional IE templates similar to those shown in Figure 2.2. Chapter 8 outlines how to improve the specification of IE templates by using SPARQL.

Query 3.1 (*SPARQL query*)

Select query requesting full names and descriptions about German women in politics with first name "Angela".

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX yago: <http://dbpedia.org/class/yago/>

SELECT DISTINCT *
FROM <http://dbpedia.org/>
WHERE {
            ?uri  rdf:type          yago:GermanWomenInPolitics;
            foaf:givenName  "Angela"@en;
            rdfs:comment    ?comment;
            foaf:name       ?name.
  FILTER (lang(?comment)="en")      }
```

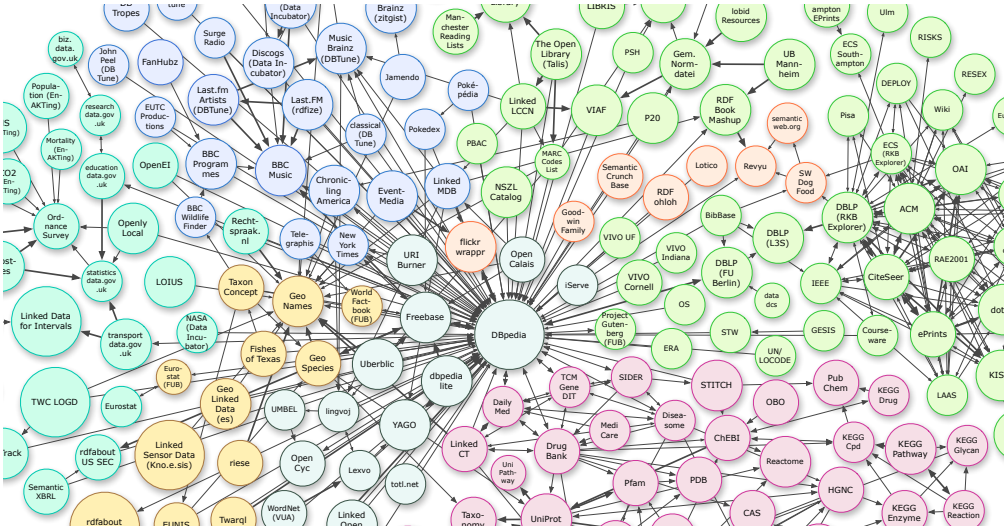


Figure 3.5: LOD cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

3.9 Utilizing RDF from the Web

The exploitation of published RDF data on the Web in IE systems is a contribution of this work. Currently, the Semantic Web is referred to as Web of data that hosts arbitrary RDF sources about various domains of concerns. Here, Linking Open Data (LOD) is a Semantic Web best practice that provides means for publishing RDF data on the Web [W3C, 2010a]. The LOD community project encourages content providers to publish data in terms of RDF by following simple guidelines [Bizer et al., 2009a]. Berners-Lee [2010] defines four recurring rules for publishing data:

Definition 3.4 (*The mantra of Linking Open Data*)

1. *Use URIs as names for things.*
2. *Use HTTP URIs o that people can look up those names.*
3. *When someone looks up a URI, provide useful information, using RDF.*
4. *Include links to other URIs, so that they can discover more things.*

LOD is a treasure cove for RDF-based IE applications. The LOD cloud in Figure 3.5 gives an impression how much data exists that can be used for IE purpose.

3.10 Unsolved and emerging challenges

The Semantic Web is an emerging bundle of technologies offering methods to represent, publish, and consume formal information on the Web. Here, the grounding challenge is to automatically translate existing information from plain Web pages into a machine-interpretable form.

Despite of the production of formal information, the consumption of published RDF data is complicated by the low quality [Jain et al., 2010]. When utilizing RDF into IE, the issues that turn out to be of major importance are illustrated in the following list and elucidated by using examples from the DBpedia:

Ambiguities between instances Frequently, different instances share a similar set of labels. This induces ambiguity problems as they occur in interpreting natural language. For example, the term "Gold" is assigned as `rdfs:label` by a large list of different instances in DBpedia[2] covering a range from music albums, minerals, movies, or results in competitions. These ambiguities result from label homonyms (e.g., the band "Gold", and Radio station "Gold") or overgeneralized label associations (e.g., labeling the career of Barack Obama[3] as "Barack Obama").

Conversely, ambiguities occur if a single entity referent is referred to by multiple instances, e.g., `dbp:Category:Barack_Obama` and `dbp:Barack_Obama` that share similar labels.

Bad labels of instances Very often labels of instances are unlikely to occur as entity references in text. For example, bracket suffixes such as *"Harry Potter (film series)"* are a common label pattern in DBpedia. Some labels consist of additional descriptions, e.g., *"Harry Potter and the Goblet of Fire Movie Poster Book [With Posters]"* is a label of a RDF represented product of the Amazon online store.

Under-specifications of datatype properties Neither RDF, RDFS, nor OWL provide means for formalizing datatype properties as describing proper names or additional attributes values such as length or height. In consequence, the whole set of datatype properties used within an RDF graph has to be analyzed before any of these properties' values may be used with entity recognition algorithms.

Incomplete models In general, RDF knowledge bases do not provide a complete coverage of used vocabularies for describing facets of represented instances. For example, although DBpedia uses the FOAF vocabulary and FOAF provides properties representing a full name, nick name, first name and family name, instances often just possess values for full names.

[2] `http://dbpedia.org/sparql?query=select+distinct+*+where+{%3Fs+%3Fp+%22Gold%22}`
[3] `dbp:United_States_Senate_career_of_Barack_Obama`

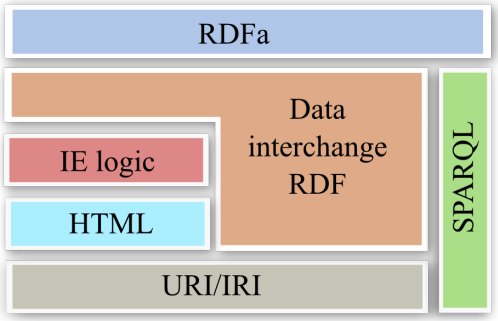


Figure 3.6: RDF-based IE-layer cake. Extraction logic is deployed as middleware component providing an RDF view on HTML data.

Inconsistent usage of vocabularies Especially in knowledge bases consisting of multiple vocabularies, overlapping properties or classes are sometimes assigned differently to single instances. For example, `dbp:Person` and `foaf:Person` are defined to be equivalent, but in DBpedia some instances are only classified by using the FOAF type.

Representation errors Especially, in automatic generated RDF knowledge bases, such as DBpedia, errors occur like assigning literal values to object properties, merging multiple different instances in a single URI, or creating nonsense properties.

The implementations of approaches that preprocess RDF data as well as processing approaches that utilize RDF data for extracting information are presented in Chapters 6 and 7. Both chapters figure out how to cope with these challenges.

3.11 Contributions

The application of RDF to Information Extraction (IE) is related to the design goal of exchanging information on the Web. Knowledge on the Web is represented in RDF. Hence, conforming IE on the Web returns extracted information in RDF format, either. Figure 3.6 outlines a modified version of the bottom part of the Semantic Web layer cake. It declares IE as middleware between HTML content and RDF data.

- The specification of RDF intends to provide information interoperability on the Web in a standardized data format. The utilization of such RDF encoded information in IE applies this intention in terms of the Hypotheses H.1 Further details on the algorithmic usage of RDF are described in Chapters 5, 6, and 7.

- Hypothesis H.2 confirms with the Semantic Web use case that a middleware should provide RDF views about Web resources. In fact, the serialization of IE results in RDF described in Chapter 8 provides exactly such a perspective on a natural language text. In addition, the use of SPARQL Query Language (SPARQL) enables people to employ tokens of the vocabulary underlying the data of concern as filters in SPARQL queries describing what information the IE-system must extract from text.
- RDF is a data format, capable to represent any kind of domain knowledge. Hence, providing a support for incorporating RDF data in IE confirms the claim for adaptability in the Hypothesis 3 (see Chapters 5 and 6).

4 Ontology-based Information Extraction

> "Automatic metadata generation would be the snowball to unleash an avalanche of metadata through the Web, making the Semantic Web come true."
>
> *(Popov et al. [2004])*

Popov et al. [2004] stated that the identification and extraction of formal knowledge in text bears great potential. So far, IE-results have not being provided sufficient degrees of completeness, clarity, and relevance. The reason for this is figured out by Brewster et al. [2003]: *"An ontology reflects the background knowledge used in writing and reading a text. However, a text is an act of knowledge maintenance, in that it re-enforces the background assumptions, alters links and associations in the ontology, and adds new concepts. This means that background knowledge is rarely expressed in a machine interpretable manner."* Consequently, Buitelaar et al. [2005] conclude that most of the knowledge in text is implicit and remains "under the surface". The approaches proposed in this work provide information extractors with formal domain knowledge for supporting the correct semantic interpretation of entity references.

This chapter surveys literature about utilizing ontologies in IE-systems and distinguishes this work from related approaches. First, Section 4.1 introduces a general Ontology-based Information Extraction (OBIE) architecture and separates between ontology learning and ontology population. Second, Section 4.2 describes existing IE-systems that utilize ontologies. Extraction ontologies as special form of integration are addressed in Section 4.3. Next, Section 4.4 figures out the challenges current OBIE system have to cope with. Contributions to these challenges are exposed in Section 4.5.

4.1 Ontologies in Information Extraction

The intention of this work addresses the utilization of formally represented information in IE-systems. In literature, this is referred to as Ontology-based Information Extraction (OBIE). Embley et al. [1998] coined this term in an IE approach for populating the content of a relational database. Here, the database is described as kind of ontology. Later on, OBIE was further promoted by Bontcheva et al. [2004] and integrated into the General Architecture for Text Engineering (GATE). The *Workshop on Ontology-based Information Extraction Systems* (OBIES 2008) [Adrian et al., 2008b] helped to establish the discrete nature of Ontology-based Information Extraction (OBIE).

Wimalasuriya and Dou [2010] summarize the state-of-the-art in OBIE and defined corresponding systems as follows:

Definition 4.1 (*An ontology-based Information Extraction system*)

"A system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies." Wimalasuriya and Dou [2010]

In terms of OBIE Wimalasuriya and Dou [2010] separates between two general approaches:

Ontology learning approaches construct an ontology by processing natural language text. The term was originally coined by Maedche and Staab [2001]. Cimiano [2006] describes ontology learning as acquisition of a domain model from data. He illustrates a process (shown in Figure 4.1) about the learning of ontologies from text, which involves NLP techniques. It is divided into following steps:

- Extraction of domain terminology and synonyms from a corpus of documents (e.g., {city}, {country, nation})
- Identifying of main concepts on the basis of the detected relevant terms and the classes of synonyms (e.g., $c :=$ country $:=$ nation)
- Structuring of concepts into taxonomy (e.g., capital $<$ city)
- Learning of non-taxonomic relations between concepts (e.g., is_capital(city, country))
- Structuring of relations into hierarchy(e.g., is_capital(city, country) ¡ located_incity, country)
- Learning the adoption of axiomatic definitions of and between concepts (e.g., disjoint, equivalence) and relations (e.g., cardinalities, transitive, reflexive, symmetric)
- Learning general axioms (e.g., currentProject(person, project) $\wedge$ member(project, organization) $\rightarrow$ member(person, organization))

Related to learning ontologies, Buitelaar et al. [2005] remarks that *"an ontology is a view on how the world or a specific domain is structured as agreed upon by the members of a community"*. Hence, he concludes, *"ontologies as such cannot be learned by machines in the strict sense of the word"*. The reason is that *"machines lack intention and purpose"*. *"Instead, ontology learning techniques support ontology engineers"* ... *"on the basis of empirical evidence derived from textual or other data"*.

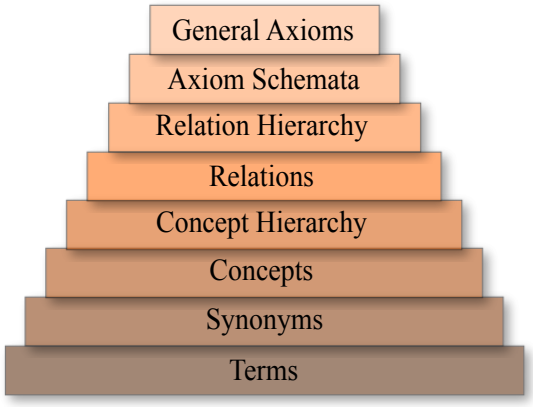


Figure 4.1: Ontology learning stack, by Cimiano [2006].

Ontology population approaches identify entities in text that are related to a predefined domain ontology. Cimiano [2006] outlines three major task topics in populating ontologies:

- Learning instances of concepts in the domain ontology (e.g., concrete cities, persons, or locations). This task is similar to a NER. Here, the formal concept definition makes the difference.
- Learning instances of formalized relations between two or more instances of concepts (e.g., instantiations of the relationship `foaf:member` between an instance of `foaf:Group` and any kind of `foaf:Agent` such as a `foaf:Person`.)
- Semantically annotating entity references with instantiations of relations or concepts in a domain ontology. This topic is going to be discussed in more detail in Chapter 5 where it is referred to as *Semantic Entity Recognition*.

Wimalasuriya and Dou [2010] stated that most OBIE systems only extract instances and property values with respect to classes and properties of a given ontology. Here, ontology population is implemented as set of information extractors, each extracting individuals for a class or property value for a property. This statement is applicable

to the intended goals of this work, which investigate ontology population techniques. In terms of OBIE, the research objectives of this work follow Li and Bontcheva [2007], who described OBIE as: *"use the representation of formal ontologies in Information Extraction as one of the system inputs and as the target output"*. Wimalasuriya and Dou [2010] categorized this characteristic of OBIE as: *"Present the output using ontologies"*.

With respect to the statement of Popov et al. [2004], about the impact of OBIE as kind of automatic metadata generation, Wimalasuriya and Dou [2010] outline a close relation between OBIE and the Semantic Web. OBIE systems create semantic content for the Semantic Web, which semantic agents can directly process. This is conform with the application of OBIE in Figure 3.6 presented in Section 3.11.

With respect to its application in IE Wimalasuriya and Dou [2010] discuss the value of an ontology as:

> *"A given domain ontology can be successfully used by an OBIE system to extract the semantic content from a set of documents related to that domain, it can be deduced that the ontology is a good representation of that domain. Further, the weaknesses of the ontology can be identified by analyzing the types of semantic content it has failed to extract."*

This relates to the challenges listed in Section 3.10, which outline possible anti-patterns in ontologies when utilizing the provided knowledge in information extractors.

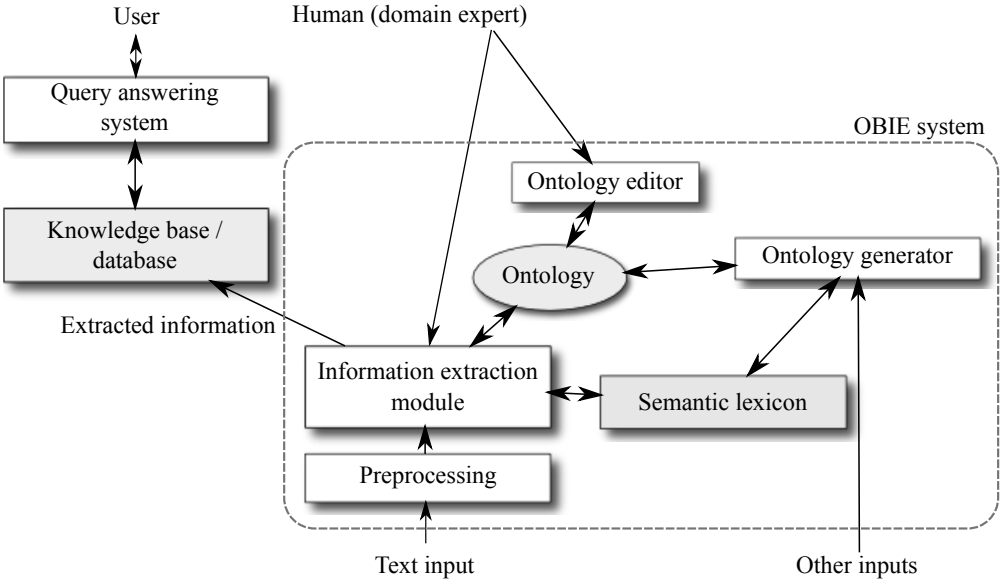


Figure 4.2: The OBIE system, by Wimalasuriya and Dou [2010].

4.2 Ontology-based Information Extraction Systems

Figure 4.2 presents possibilities of combining ontologies with IE systems. It separates between a knowledge base and an ontology. The knowledge base is queried by users and populated by information extractors. The ontology is utilized by these IE modules. Similar to the claim in the Hypothesis H.2, Wimalasuriya and Dou [2010] generalize IE templates as traditional input devices by applying a formal knowledge base. The IE modules as such are enhanced with ontologies as well as semantic lexicons. The separation between an ontology and a semantic lexicon splits the knowledge representing a domain of concern from the more general linguistic knowledge about a language in lexicons. The creation of both, ontological domain knowledge and lexicons may be supported by external sources. Interestingly, this architecture allows human domain experts to influence internal extraction logic, explicitly.

The research objectives of this work confirm with Wimalasuriya's architecture in Figure 4.2. More specifically, the information extractors in this work utilize RDF graphs from Semantic Web sources, extract information from text, and return extracted information as RDF graphs. SPARQL queries provide means for additional filtering mechanisms. The role of semantic lexicons is fulfilled by applying linguistic resources such as text segmenters, POS taggers, and text chunkers.

The following section provides an overview on IE, annotation and ontology population systems that relate to the research objectives of this work. The distinctions between these approaches and the approaches presented in this work will be discussed in Chapters 5, 6, and 7 in more details:

- The GATE framework [Bontcheva et al., 2004] enables the use of ontologies in IE by providing *OntoGazetteers* and *OntoRootGazetteer*. OntoGazetteers allow a manual mapping between gazetteer lists to ontology classes. OntoRootGazetteer analyze existing concept labels in ontologies with tokenizers, POS taggers, and stemmers in order to recognize these labels in text sources.

- Many OBIE systems are employing the technology provided by GATE. Li and Bontcheva [2007], Popov et al. [2003], and Saggion et al. [2007], for example, apply GATE and labels inside ontologies for implementing instance resolution tasks, similar to the approach presented in Section 7.4.

- Independently from GATE, Embley et al. [1998] and Sintek et al. [2001] describe a use case for populating domain ontologies with results from IE-systems. Buitelaar et al. [2006] propose SOBA, a scenario for integrating an ontology about soccer and IE-approaches to extract soccer results from semi-structured Web pages.

- In S-Cream, Handschuh et al. [2002] enriched the content of Web pages with semantic annotations originating from domain ontologies, semi-automatically. The

interesting fact about S-Cream is that it used an IE-system without ontology support. As consequence, the authors complain about issues in aligning IE-results to particular parts of the domain ontology.

- Endres-Niggemeyer [2008] proposed the use of distributed semantic agents as an architecture for interacting IE-modules. Within this application scenario, the agents utilize a shared domain ontology for extracting information from medical documents.
- The Firefox plug-in Piggy Bank extracts information from Web pages by using manually created information extractors called screen scrapers. Results are stored in a local or a global RDF store [Huynh et al., 2007]. A screen scraper is a Javascript program that extracts RDF information from within a Web document's content.
- Zemanta [Tori, 2008] is a Web service for building Web mashups. Zemanta also spots for labels of DBpedia or Freebase[1] resources (namely instances) in Web pages. The API returns results in RDF format and provides ratings about relevance and certainty.
- Open Calais[2] provides services for instance resolution. The focus of Open Calais is set on News content. Instances retrieved by Open Calais are defined in a proprietary ontology.
- Ontos' Semantic API[3] is similar to Open Calais' services. Ontos also hosts a proprietary ontology populated with instances that can be retrieved ass mentions in text.
- The providers of the DBpedia knowledge offer an instance resolution in the DBpedia Spotlight service Mendes et al. [2011]. The implementation is focused on the DBpedia knowledge base.

Section 9.1.2 compares results from DBpedia Spotlight, Open Calais, and Zemanta with the SCOOBIE system.

4.3 Extraction ontologies

In contrast to the ontology population approaches listed above, the approach of *extraction ontologies* applies a special ontology type to support information extractors along an extraction pipeline. Embley et al. [2002] coined extraction ontologies in an approach

[1] http://www.freebase.com
[2] http://www.opencalais.com
[3] http://www.ontos.com

that formally defined wrappers as ontologies to extract data from source records into a target schema. They described such an extraction ontology as follows:

Definition 4.2 (*Extraction ontology.*)

*"An extraction ontology is a conceptual-model instance that serves as a wrapper for a narrow domain of interest such as car ads. The conceptual-model instance includes objects, relationships, constraints over these objects and relationships, and descriptions of strings for lexical objects and keywords denoting the presence of objects and relationships among objects."*Embley et al. *[2002]*

The intention of extraction ontologies is the formal declaration of IE templates in order to facilitate the adaption of an IE system to a specific domain. As shown in Figure 4.3, Embley et al. [2002] applied extraction ontologies for extracting information from HTML tables about car offers in a car ads domain.

```
Car [-> object];                    // Definition of car ad object
Car [0:1] has Year [1:*];
Car [0:1] has Model [1:*];
Car [0:1] has Mileage [1:*];
Car [0:1] has Price [1:*];

PhoneNr [1:*] is for Car [0:1];     // Relating a phone number to
                                    // a car ad
Year matches [4]                    // Syntax definition of year
 constant { extract "\d{2}";
            context "\b'[4-9]\d\b";
            substitute "^" -> "19"; },

Mileage matches [8]                 // Defining keyword evidences
  keyword "\bmiles\b", "\bmi\.", //  to milage references
  "\bmileage\b", "\bodometer\b";
      ...
```

Figure 4.3: Extraction ontology adding lexical pattern definitions to a conceptual modeling of a car ads domain [Embley et al., 2002].

Labsky [2008] extended the approach of extraction ontologies by specifying the Extraction Ontology Language (EOL). EOL is an XML-based language used to encode extraction ontologies.

4.4 Unsolved and emerging challenges

This chapter revealed that the unsolved problems of OBIE, ontology learning, or ontology population remain on a rather conceptual level. Although Wimalasuriya and Dou [2010], Nedellec and Nazarenko [2006], Cimiano [2006], Buitelaar et al. [2005], and Maedche [2002] provide comprehensive overviews on the use of ontologies in IE, they agree on the fact that, even with the support of ontologies, automatic text understanding and therefore IE is still an unsolved problem field. By studying different ontology-based approaches, it can be seen that no consensus exists about how to support IE at best by applying ontologies or any kind of formal background knowledge. Hence, improving the effectiveness of the IE process by incorporating formal background knowledge is still an open issue. Nevertheless, the above mentioned authors agree about the huge potential and influence the integration of OBIE systems with the Semantic Web bears.

4.5 Contributions

In terms of OBIE and ontology population, results of this work bear the following contributions:

- Intended by the Hypothesis H.1, the Semantic Entity Recognition process in Section 5.3 extends the description of OBIE tasks. Here, existing IE tasks are refined by the use of formal background knowledge:

  1. Recognizing instantiations of concepts in a text (see Section 7.3).
  2. Annotating entity references with such instantiations (see Sections 7.4, 8.4).
  3. Recognizing instantiations of formal relations (see Section 7.8).

  In addition to these tasks, an extended list of OBIE-enabled tasks are defined:.

  1. Filter segments in larger amounts of textual content that might be candidates for being entity references to instantiations of formal concepts (see Section 7.2).
  2. Separate the identification of instantiations of formal datatype properties from formal object properties in a text (see Section 6.2).
  3. Disambiguate multiple referents of instances caused by unclear entity references (see Section 7.5).
  4. Rank recognized concept instances by relevance criteria.(see Section 7.6).

  Solutions to these knowledge-processing issues are described in more detail in Chapter 7.

- Hypothesis H.2 demands an integration of OBIE into the Semantic Web technology stack consisting of URIs, RDF, and SPARQL. Despite DBpedia Spotlight, which is specialized to the DBpedia domain, none of the presented OBIE system implements this integration. Chapter 8 proposes methods for serializing IE results in RDF by preserving the terms of the underlying domain of concern. It also presents the use of SPARQL for specifying OBIE templates.
- In contrast to the presented approaches, which utilize proprietary or specialized ontologies, the approaches being presented in Chapter 6 apply to any kind of ontologies stating they are represented in RDF. Compared to extraction ontologies, the approaches presented in this work do neither require any modifications of existing ontologies nor do they require the creation of handcrafted representation devices or IE rule sets (refer to the Hypothesis H.3).

5 Foundations for utilizing RDF in Information Extraction

> A proper name is a word that answers the purpose of showing what thing it is that we are talking about but not of telling anything about it.
>
> *(John Stuart Mill, 1843, A System of Logic, Ratiocinative and Inductive)*

References to real world entities occur in natural language text as well as a in formal ontologies. Hence, combining natural language with formal ontologies involves associations between both reference types if they correlate with the same conceptual referents. Technically, this work utilizes RDF in IE by recognizing URI references in RDF graphs that correspond with proper names in text.

This chapter outlines the values of incorporating RDF graphs into information extractors. First, Section 5.1 introduces Peirce's formal sign relations to reveal correlations between knowledge represented in natural language and in Resource Description Framework (RDF). This investigation results in extending named entities, as they are defined and used in IE, to become semantic entities. Next, Section 5.2 lists components of the RDF model and analyzes the utilization within information extractors. As a result, the Semantic Entity Recognition process is defined in Section 5.3. Finally, Section 5.4 summarizes techniques for preprocessing knowledge in RDF as well as extracting information from text in scope of the Semantic Entity Recognition process.

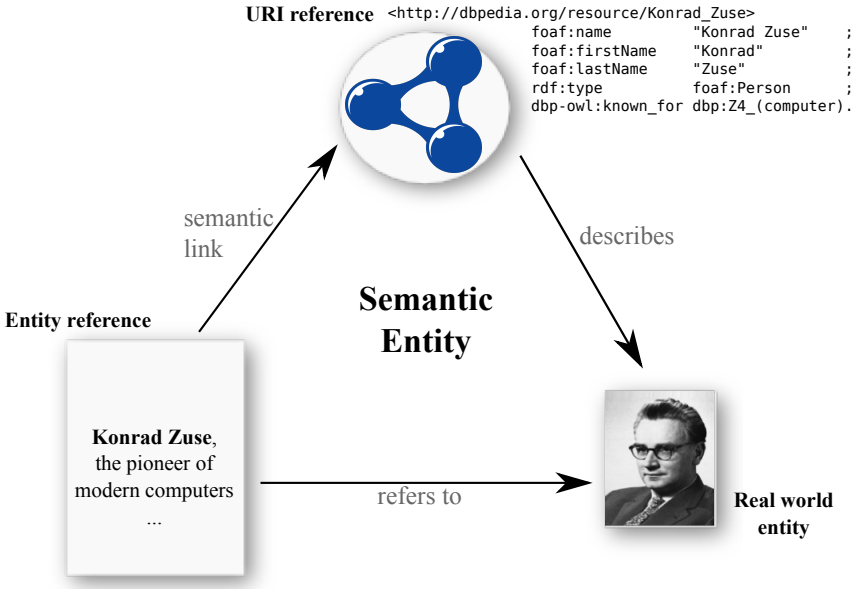


Figure 5.1: Triadic co-relation between entity references and URI references.

5.1 Linking URIs and textual references

Within natural language text, references to real world concepts are made by mentioning proper names (e.g., "Konrad Zuse") or by paraphrasing the behavior of concepts with attributes ("Inventor of Z3"). In Computational Linguistics, these references are defined as *entity references* (short entities). If a *proper name* in text is determined as entity reference, it is referred to as a *named entity* in literature. Similar to referring to entities from text, in RDF instances, which are named by URIs, refer to real world concepts. The scientific foundation behind such a co-reference is provided by Peirce [1976], who defined a theory about sign relations:

> *"Namely, a sign is something, A, which brings something, B, its interpretant sign determined or created by it, into the same sort of correspondence with something, C, its object, as that in which itself stands to C."*

Figure 5.1 applies this triadic relationship to entity references in the natural language text (A), the instance in the RDF graph (B), and finally the corresponding real world concept (C). On the base of this triadic relationship, it can be concluded that utilizing RDF data in Named Entity Recognition (NER) (as described in Section 2.2.2) requires the alignment of named entities with corresponding instances. Therefore, the traditional NER approach is extended to also perform a *Semantic Entity Recognition*, which aligns *named entities* with URI references. More concretely, the recognition of semantic entities

creates links between entity references in the text and instances in the RDF graph if both references target to exactly the same real world concept. In the remainder, these bridges between natural the language text and the RDF graph are from now on referred to as *semantic links*.

Definition 5.1 (*Semantic link, semantic entity*)

*A **semantic link** is an explicit association between a named entity NE and an instance URI, if NE and URI resolve to the same referent. Any named entity that possesses a semantic link is referred to as **semantic entity**.*

In terms of an actual concept that is referred to from text, the existence of a semantic link to a formal instance allows the incorporation of additional information about this instance from RDF graphs.

5.2 RDF components

The utilization of RDF provides information extractors with additional information on semantic features about entities. This information exceeds the syntactic interpretation capabilities of traditional NER approaches. The value of RDF components is presented for being used by information extractors.

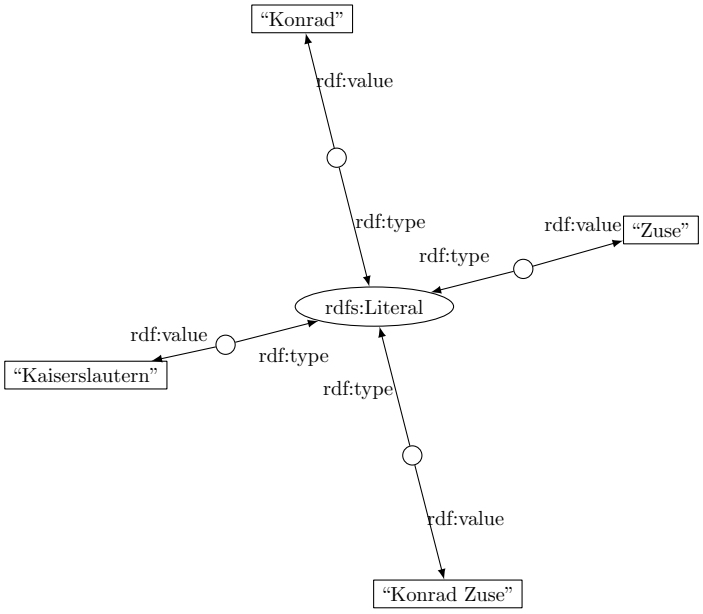
5.2.1 Literals

Literal values in RDF graphs determine property values such as weight, height, dates, and prizes of instances. Literals also comprise proper names or structured identifiers, such as ISBN numbers.

As shown in Figure 5.2, the recognition of a match between an RDF literal and an entity allows the definition of single text segments as literals of an RDF graph (i.e., “Kaiserslautern”, “Konrad Zuse”, “Konrad”, and “Zuse”). As consequence, text segments matching with RDF literals can be considered as relevant in terms of the RDF graph’s domain of concern.

5.2.2 Datatype properties

In RDF, datatype properties determine the semantics of literal values. Applied to semantic entities in the natural language text, datatype properties attach a formal semantics to the associative meaning of semantic links. Figure 5.3, for example, describes the entity “Konrad” to match with a literal value of the `foaf:firstName` property. `foaf:firstName` describes the first name of an instantiation of a `foaf:Person`.



"Konrad Zuse has never been to Kaiserslautern."

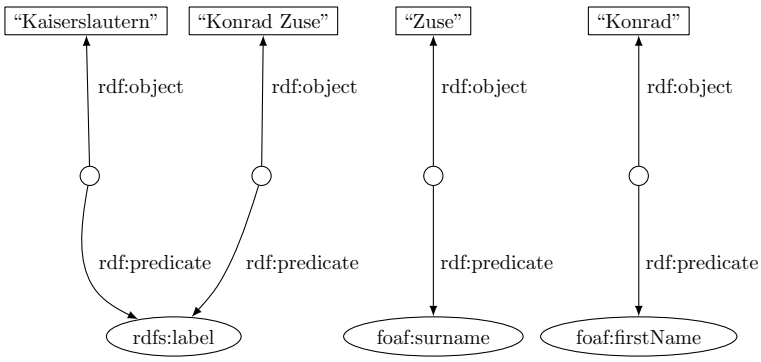
Figure 5.2: Markup text segments as formal literal values of an RDF graph.

The semantic value of recognized datatype properties depends on how restrictive its signature is defined within the formal vocabulary. As shown in the listing below, the Friend of a Friend (FOAF) vocabulary declares instances with `foaf:firstName` properties to be classified as `foaf:Person`.

Example 5.1 (*Signature of `foaf:firstName`*)

```
foaf:firstName  rdfs:domain  foaf:Person ;
                rdfs:range   rdfs:Literal .
```

Hence, the entity "Konrad", which is recognized as a property value of `foaf:firstName`, can now be inferred to relate to an instance of type `foaf:Person`. In contrast to FOAF's `foaf:firstName`, the RDFS vocabulary defines the property `rdfs:label` more loosely as being a human-readable version of a resource's name [Brickley and Guha, 2004].



“Konrad Zuse has never been to Kaiserslautern.”

Figure 5.3: Assign RDF datatype properties to recognized entities.

Example 5.2 (*Signature of* `rdfs:label`)

```
rdfs:label  rdfs:domain  rdfs:Resource  ;
            rdfs:range   rdfs:Literal   ;
            rdf:type  owl:AnnotationProperty  .
```

The formal OWL definition of `rdfs:label` classifies it as `owl:AnnotationProperty`, which determines annotating properties whose values should be used as a basis for subsequent inferences [Bock et al., 2009]. The recognition of the `rdfs:label` “Kaiserslautern” does not produce any hints for inferring any further classifications. In summary, the recognition of datatype properties of semantic entities is required to infer further classifications, but does not imply these in all cases. Additional information about instances has to be utilized from RDF graphs.

In general, datatype properties formally subsume a shared nature of a list of literal values. This relates to gazetteers in traditional NER (see Section 2.2.2) consisting of lists of first names, academic title abbreviations, or currency symbols.

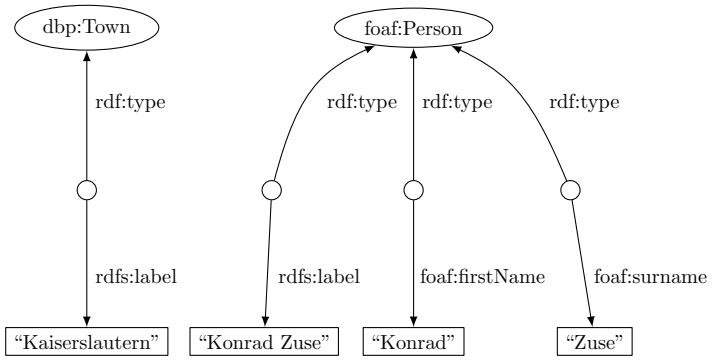
5.2.3 Types

The example in Figure 5.3 shows that literals of the semantic entities “Kaiserslautern” and “Konrad Zuse” are values of the datatype property `rdfs:label`. As a `rdfs:label` value is defined to determine only a name of something, it is impossible to infer additional information about the instances both entities refer to. Supporting, for example, a Word-sense Disambiguation requires additional information, such as the instances’ types. In Figure 5.4, the type of “Kaiserslautern” and “Konrad Zuse” can be resolved by interpreting the values of the `rdf:type` property, i.e. `foaf:Person` and `dbp:Town`.

Example 5.3 (*Explicit classification knowledge by `rdf:type` statements.*)

```
[] rdfs:label 'Kaiserslautern' ;
   rdf:type dbp:Town .
[] rdfs:label 'Konrad Zuse' ;
   rdf:type foaf:Person .
```

Adding `rdf:type` information to a list of datatype property values leads to a formal representation of gazetteers populated with names of respective types, such as names of instantiations of `dbp-ont:Town`, `dbp-ont:Country`, or `dbp-ont:Company`.



"Konrad Zuse has never been to Kaiserslautern."

Figure 5.4: Disambiguating word senses by explicit `rdf:type` statements.

5.2.4 Instances

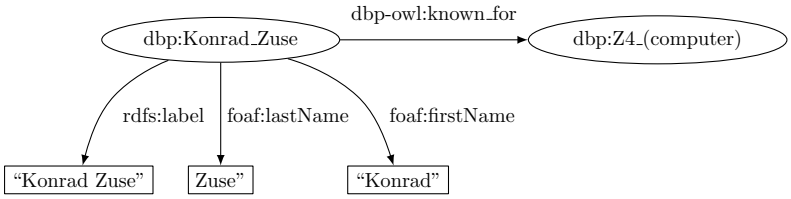
A semantic link is most valuable when it associates a named entity in text with an instance in the RDF graph. This allows the utilization of known properties about this actual instance in the RDF graph, comprising datatype property values, classifications, and relationships to other instances.

In Figure 5.5 the semantic entity "Konrad Zuse" refers to the instance `dbp:Konrad_Zuse` described by the DBpedia as `dbp-ont:Person` who is known for the "Z4" and who possessed the first and last name "Konrad" and "Zuse".

Links between entities and instances connect the formal semantics within RDF graphs to unstructured text. In terms of Computation Linguistics, this allows information extractors operating on the RDF graph for solving entity resolution problems, such as:

- The Word-sense Disambiguation of an entity on a literal value that may be resolved with multiple instances (see also Section 7.5).

- The unification of recognized entities on different literal values that all refer to the same instance in the RDF graph (see also Section 7.4).



“Konrad Zuse has never been to Kaiserslautern.”

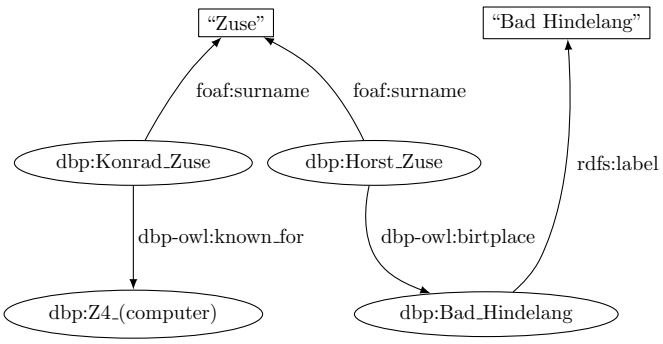
Figure 5.5: Semantic link between a named entity and a URI reference.

5.2.5 Object properties

Object properties associate two instances with an explicit relationship, which may be specified by an underlying vocabulary. The interpretation of such relational knowledge may provide a word sense resolver with evidences about why two entities co-occur in a text. For example, in Figure 5.6 the relation `dbp-owl:birthplace` associates `dbp:Horst_Zuse` with `dbp:Bad_Hindelang` meaning that Horst Zuse was born in Bad Hindelang. Here, the resolution of the entity “Zuse” is still ambiguous. Both entities may be resolved with two different instances. “Zuse” may be resolved either as name of a person called “Horst Zuse” or as name of a person called “Konrad Zuse”. However, having access to the “birthplace” relationship the resolver is enabled to resolve “Zuse” by taking into account the co-occurrence of “Zuse” and “Bad Hindelang” in text to match it with the known relationship. As result, this evidence is used to decide that, most likely, “Zuse” refers to `dbp:Horst_Zuse`.

5.3 Semantic entity recognition process

Finally, for developing and evaluating a Semantic Entity Recognition, the overall entity recognition process is divided into a sequence of information extractors. This definition allows a proper utilization of RDF within each information extractor. Each information extractor is categorized in two ways: Information extractors that pre-process knowledge in RDF graphs or text corpora. Information extractors that process the information from a text content in order to recognize semantic entities. Figure 5.7 illustrates this architecture. The following lists summarizes IE preprocessing and processing topics this



"Zuse first saw the light in Bad Hindelang."

Figure 5.6: RDF resources are interlinked with object properties.

work contributed to (more details on preprocessing and processing techniques will be provided in Chapters 6 and 7). Specific techniques that are used to implement information extractors in (pre-)processing steps will be subsequently described in Section 5.4. Chapter 8 explains details on post-processing methods that cover further processing-steps of extraction results.

Preprocessing steps consolidate and condense knowledge provided by RDF graphs and text corpora. The analyses performed by these steps are independent from the given text in progress of the entity recognition pipeline. In general, preprocessors create representational or statistical models that are later on utilized by subsequent information extractors. More details on preprocessors will be presented in Chapter 6.

The following list summarizes the investigated pre-processing approaches:

1. **A relational model of RDF data** (Section 6.2): RDF is designed to exchange formal information on the Web. Its data model is not suitable for handling retrieval, partitioning, or mining operations efficiently. Hence, a *relational database schema* was developed to store RDF graphs with focus on the requirements demanded by information extractors along the Semantic Entity Recognition process.

2. **Clustering correlating classes in RDF graphs** (Section 6.3): In general, RDF graphs classify instances by a single or by multiple class hierarchies. These hierarchies may overlap or consist of over-specified class definitions that cannot be reconstructed from the data in text. Hence, a clustering is performed to reduce the number of classes by subsuming correlating parts of hierarchies into single clusters. *Hierarchical Clustering* (Section 5.4.3) and *Principle Component Analysis* (Section 5.4.5) were applied to collapse correlating RDFS classes.

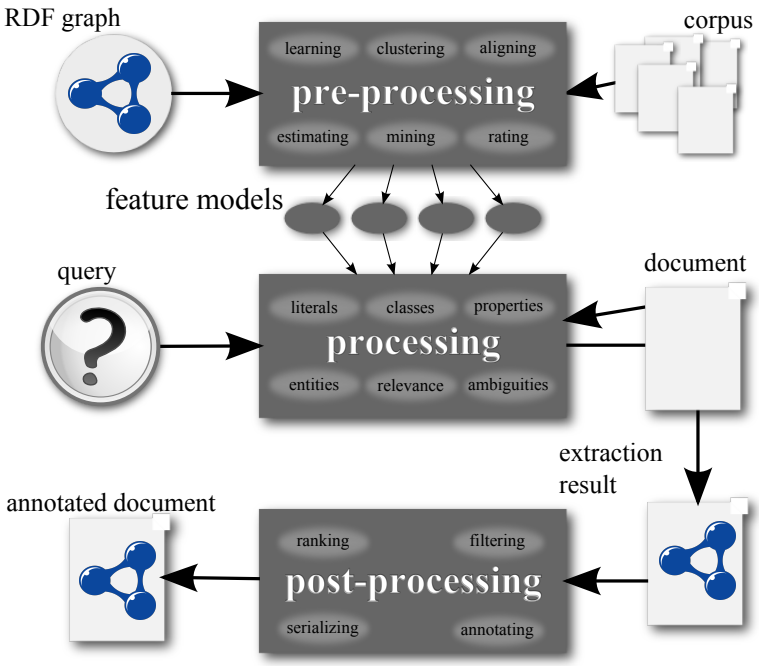


Figure 5.7: RDF-based Information Extraction architecture.

3. **RDF graph statistics** (Section 6.4): This pre-processing computes a variety of statistics about the structure in RDF graphs, such as frequency distributions of datatype properties, object properties, classes in RDF statements, and instances.

4. **Text corpus statistics** (Section 6.5): *Word frequencies* in text corpora are computed in this pre-processing step, such as the number of word occurrences within a single text (term frequency) and the number of documents that contain a word (document frequency).

5. **Mining datatype properties for proper names** (Section 6.6): *Frequency distributions of literals* in text corpora and RDF graphs are used to create proper metrics for describing RDF properties that represent proper names.

6. **Aligning datatype properties with regular expressions** (Section 6.7): Analyzes which datatype properties contain values that match with given regular expressions.

7. **Automatically labeling a text corpus with classes** (Section 6.8): Training data is necessary for training machine-learning models to automatically classify entities with RDFS classes. This pre-processing step automatically generates such training data based on a given text corpus and results of the Semantic Entity Recognition process.

Processing steps implement information extractors of the Semantic Entity Recognition. These extractors incorporate background knowledge from RDF graphs and text corpora in terms of models, which were created during the preprocessing phase. More details on information extractors will be presented in Chapter 7.

1. **Filtering text for proper names** (Section 7.2): In general, the recognition of RDF instances in text is built upon spotting values of proper name properties. Here, the recognition of noun phrases reduces the amount of comparisons. A CRF (which is described in Section 5.4.8) is trained to detect phrasal language segments in text. Detected noun phrases determine possible candidates to co-occur in RDF graphs as values of datatype properties.

2. **Spotting text for datatype property values** (Section 7.3): Recognized proper name candidates in text are compared with literal values of datatype properties that were classified to represent proper names. (This refers to the RDF graph statistics). A suffix array (which is described in Section 5.4.1) is used as technique for comparing lists of noun phrases and literal values. The relational model of RDF data is used for querying the RDF graph for potentially matching literals. This comparison results in a list of matches consisting of literal values typed as datatype properties.

3. **Linking named entities to formal instances** (Section 7.4): Based on recognized literal values and datatype properties, the instance recognition resolves candidates for instances that exist as subject in RDF triples holding the actual datatype property as predicate and literal values as object. The relational model of RDF data facilitates this resolution of subjects.

4. **Resolving ambiguous semantic entities** (Section 7.5): Ambiguously recognized instances are analyzed by using graph metrics describing the connectivity. Here, the link mining algorithms HITS and PageRank (described in Section 5.4.6) as well as standard metrics are applied to resolve the set of ambiguously recognized instances.

5. **Rating relevance of semantic entities in text** (Section 7.6): Recognized entities are rated by relevance related to the general context, which is determined by the information contained in text and RDF graph. Besides term-based corpus statistics (described in Section 6.4) again, HITS and PageRank are applied.

6. **Classifying semantic entities** (Section 7.7): A classifier predicts RDFS classes from the RDF graph for recognized proper names. The maximum entropy approach (described in Section 5.4.7) was chosen as model for classification.

7. **Predicting object properties between semantic entities** (Section 7.8): Based on the sub graph consisting of recognized instances and their object property values, this

approach predicts additional relations between instances that have not been explicate in the RDF graph, yet. Here, matrix representations (described in Section 5.4.2) are used to calculate with correlations, distances, and similarities.

5.4 Required technological fundamentals

The implementation of preprocessing and processing steps involves the application of algorithmic, statistical, and general mathematical procedures. Before explaining details on each preprocessing and processing step, the remainder of this section outlines the procedures used within this work:

5.4.1 Suffix arrays

An efficient comparison between a set of RDF literals and a plain text has to be performed in a linear time complexity. This requires both sources to be transformed into a data structure that is specialized on implementing string comparisons.

A *suffix array* describes an array of suffixes sorted in lexicographical ordering. It provides support for scalable substring matching operations [Kärkkäinen et al., 2006]. This important property of suffix arrays allows sub-string matching to be implemented by searching the lookup string as a prefix value in the sorted list of suffixes. Such a lookup can be performed in logarithmic ($O(\log(n))$) complexity.

Example 5.4 (*Suffix array*)

Transforming the string "Peter, Paul, and Mary" into a suffix array produces the following results. Here, words are the basic indexing segments of this text.

```
'and Mary'
'Mary'
'Paul and Mary'
'Peter, Paul and Mary'
```

Algorithm 5.1 describes a simple implementation of constructing a suffix array. The algorithm creates a suffix array in log-linear ($O(n*\log(n))$) time complexity. Kärkkäinen et al. [2006] proposed a method for constructing suffix arrays in linear time. Instead of creating physical substrings, the implementation of a suffix array developed within this work uses references in memory that point to text passages. This ensures that the text remains only once as string value in memory.

Algorithm 5.1 (*Simple suffix array creation*)

Input: *A parameter* `text`.

Output: *A suffix array, which is a lexicographically sorted list of suffixes of* `text`.

```
def suffix_array(text) :

 suffix_array = []                  # create array structure
 words = text.split(' ')            # tokenize text by whitespaces
 last = len(words)-1;               # index of last word

 for i in range(last) :             # create and append suffix
  suffix_array.append(' '.join(words[last-i:]))

 return sorted(suffix_array, key=str.lower)
```

When spotting RDF literals in text, the maximum suffix-length can be restricted. In terms of the DBpedia, a maximum of 100 characters per literal is applied[1].

In theory, suffix arrays are often referred to as a pragmatic generalization of a suffix tree, which offers a richer set of string matching functionalities. In practice, suffix arrays are more applicable. Because of their list-like nature, it is simple to serialize suffix arrays to files. However, it is possible to process a suffix array as stream of string values without the need to store it in main memory completely. A suffix tree consists of highly interconnected pointer structures, which requires to keep the whole suffix tree in main memory. Finally, it can be stated that the suffix array provides a computable representation of text that offers efficient search features.

5.4.2 Matrix computations

Linear algebra offers various procedures for computing with multivariate data sources. In order to apply algebraic operations to formally represented information, a matrix representation is used. Information about an instance is represented within a matrix as a row. Each column of a row determines a certain property or feature of this instance. For example, the matrix M classifies the instances *Peter*, *Paul*, and *Mary* as *persons*, *males*, or *females*. In the following chapters, when referring to matrices, the index i refers to the row of a matrix, j denotes the column, m represents the count of rows, and n represents the count of columns. Indexes will be counted starting from zero.

[1] Even the longest European village name, *Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch* is just 58 characters long (see http://en.wikipedia.org/wiki/List_of_long_place_names).

Example 5.5 (*Matrix representation of instances*)

$$M = \begin{pmatrix} i\backslash j & person & male & female \\ Peter & 1.0 & 1.0 & 0.0 \\ Paul & 1.0 & 1.0 & 0.0 \\ Mary & 1.0 & 0.0 & 1.0 \end{pmatrix}$$

Such a matrix representation allows the calculation of similarities or distances between instances. *Euclidean distance* between two instances A and B is calculated as follows:

Definition 5.2 (*Euclidean distance*)

$$Euclidean\ distance(A, B) \sqrt{\sum_{j=1}^{n} (A_j - B_j)^2}$$

When interpreting instances A and B as vectors the *cosine similarity* between $\vec{A}$ and $\vec{B}$ determines the angle between both. Compared to the *Euclidean distance*, the *cosine similarity* is not influenced by the length of the vector. *Cosine similarity* values are within a range of zero and one.

Definition 5.3 (*Cosine similarity*)

$$cosine\ similarity(\vec{A}, \vec{B}) = \frac{\sum_{j=1}^{n} \vec{A}_j \times \vec{B}_j}{\sqrt{\sum_{j=1}^{n} (\vec{A}_j)^2} \times \sqrt{\sum_{j=1}^{n} (\vec{B}_j)^2}}$$

5.4.3 Hierarchical clustering

Based on a matrix populated with information on instances, the *agglomerate hierarchical clustering* algorithm [Duda et al., 2001] provides means for clustering instances by maximal similarity or minimal distance. The agglomerative clustering in Python that is presented in Algorithm 5.2 starts with a given set of instances within a matrix. During each subsequent step, the algorithm merges the closest or most similar pair of instances or clusters to a new cluster until the desired number of clusters is reached.

Algorithm 5.2 (*Agglomerative clustering algorithm*)

Input: *A* `matrix` *representing instances, and a parameter* `desired_cluster_number`*.*

Output: *A clustered matrix, consisting of* `desired_cluster_number` *columns.*

```
def agglomerative_clustering(matrix, desired_cluster_number) :

  while matrix.rows > desired_cluster_number :
    C1, C2 = matrix.find_nearest_cluster()
    matrix.merge(C1, C2)

  return matrix
```

In this work, hierarchical clustering is used to subsume similar RDFS classes (as described later in Section 6.3). Closeness between single instances is computed by the Euclidean distance. In detail, the distance between two clusters is computed as the mean distance between contained instances. This strategy is also referred to as *average linkage clustering* [Duda et al., 2001].

Definition 5.4 (*Average linkage*)

$$\textit{average linkage}(\vec{C_1}, \vec{C_2}) = \frac{\sum_{A \in C_1} \sum_{B \in C_2} \textit{distance}(A, B)}{|C_1||C_2|}$$

5.4.4 Descriptive statistics

In descriptive statistics, the column of a matrix may be interpreted as a variable X of a sample distribution of values. The statistical ratios mean, variance, covariance, and correlation provide more specific descriptions on the distribution of X. In this work, the arithmetic mean of a sample of the variable X is also interpreted as expected value (μ) of the variable X. They are defined as follows:

Definition 5.5 (*Arithmetic mean*)

The arithmetic mean denotes the average value of the distribution of X*.*

$$\textit{mean}(X) = \mu(X) = \frac{\sum_i^m X_i}{m}$$

Definition 5.6 (*Variance*)

The variance describes the deviation between the realization of X and its mean.

$$var(X) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \mu(X))^2$$

The variance of a variable X is often interpreted as the contained degree of information. Knowing the value of a variable that possesses a low amount of variance does not contain much information. For example, the knowledge about Peter being a person is not informative as all instances within the matrix are classified as person. Hence, the variance of (X = person) is 0 in M. Gender information is determined by a variance ratio of $\text{var}(X) = 1/3$.

Definition 5.7 (*Covariance*)

The covariance describes the dependence between two variables X and Y.

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \mu(X))(Y_i - \mu(Y))$$

A negative covariance value describes an anti-proportional linear dependency between variables X and Y, i.e., increasing the value of X causes a decreasing value of Y. A positive covariance denotes a proportional linear dependency. A zero value expresses the absence of any linear dependency between X and Y.

Correlation

Pearson's product momentum normalizes the covariance to produce values in a range between minus one and one. Pearson's product momentum is simply the covariance of the normalized statistical distributions of variables X and Y, which possesses a zero mean of and a standard deviation ($\sqrt{\text{var}(X)}$) of 1.0.

Definition 5.8 (*Pearson's product momentum*)

$$correlation(X,Y) = \frac{cov(X,Y)}{\sqrt{var(X)}\sqrt{var(Y)}}$$

The scatter plots illustrated in Figure 5.8[2] visualize a series of correlation values between a range from -1.0 to 1.0.

[2] originating from http://en.wikipedia.org/wiki/File:Correlation_examples2.svg, 2011-06-16

Figure 5.8: Correlation values between variables in a 2-dimensional space.

5.4.5 Principle component analysis

Similar to the hierarchical clustering the goal of applying a Principle Component Analysis (PCA) is to collapse similar properties of instances. Therefore, the matrix M $(n \times m)$, which describes m instances along n columns, should be transformed by a projection matrix T into a new matrix M_r $(n_r \times m)$ with a reduced number $n_r < n$ of columns.

Definition 5.9 (*Projection matrix*)

$$M_r = TM$$

The properties within the projected matrix M_r should possess a maximum of variance. Following Duda et al. [2001] this optimization problem can be reformulated to:

Definition 5.10 (*Eigenvalue equation*)

$$COV(M)T = \alpha T$$

This equation is satisfied if T is a matrix of eigenvectors of the matrix of covariance values of M COV(M) and α determines the corresponding eigenvalues. These eigenvectors and eigenvalues can be computed by applying the *singular value decomposition* to the covariance matrix [Wall et al., 2002]. The vector α contains the eigenvalues in decreasing order. The matrix T contains the corresponding eigenvectors. As the nature of eigenvalues corresponds with the notion of variance, the interpretation of αT allows to state that the first column vector of T determines the first principle component of the distributions of M.

Now, for collapsing invariant columns of matrix M a fixed proportion p can be calculated by a sum of eigenvalues along $k <= n$ columns:

Definition 5.11 (*Cumulative degree of informativeness*)

$$p = \frac{\sum_{\lambda_i \in \alpha}^{k} \lambda_i}{\sum_{\lambda_i \in \alpha}^{n} \lambda_i}$$

If the value of p exceeds a threshold t, k determines the number of eigenvectors to choose from T. These columns are represented within a reduced projection matrix T_r. M_r is defined and normalized to possess a zero mean and standard deviation of one by:

Definition 5.12 (*Final projection*)

$$M_r = \frac{T_r(M - mean(M))}{\sqrt{\alpha}}$$

5.4.6 Link analysis in graphs

The representational model of RDF is a graph. Therefore, *adjacency matrices* are used as representation to allow a mathematical calculation with RDF data.

Definition 5.13 (*Adjacency matrix*)

When considering a graph $G(V, E)$ to consist of a set vertexes V and of edges E in between vertexes, the ***adjacency matrix*** *is a quadratic matrix in which rows and columns represent the graph's vertexes. Boolean values within the matrix express the existence of an edge between vertex i and j whereas real values assign an edge weight between vertex i and j. An undirected graph results in a symmetric adjacency matrix.*

By adding restrictions on the sum of row values, an adjacency matrix can be transformed into a *probabilistic transition matrix*.

Definition 5.14 (*Probabilistic transition matrix*)

A normalized form of an adjacency matrix assures that the sum of entry values within each row sum up to one. In this case, each value of a matrix cell can be interpreted as probability. The adjacency matrix is then referred to as ***probabilistic transition matrix****.*

A *link analysis* within graphs intends to raise descriptive statistics about the connectivity of vertexes. Transferred to RDF graphs, such statistics allow the evaluation of importance of relatedness between a single instance and an existing RDF graph.

Node connectivity

In graph theory, several methods exist for computing a node's connectivity. Representing an RDF graph as directed graph $G(E, V)$ which represents instances as vertexes V and object properties as directed edges $E(V_s, V_o)$, allows the application of following metrics:

Definition 5.15 (*The degree of a node*)

The degree of a node V denotes the number of incoming edges plus the amount of outgoing edges.

$$degree(V) = |E(V_i, V)| + |E(V, V_j)|$$

Definition 5.16 (*The capacity of a node*)

The capacity of a node V denotes the minimum count of incoming and outgoing edges.

$$capacity(V) = min(|E(V_i, V)|, |E(V, V_j)|)$$

Two prominent link analysis algorithms, namely HITS and PageRank, are applied to RDF graphs. Both algorithms are grounding their calculations on the eigenvalue analysis of the adjacency matrix.

Hyperlink-Induced Topic Search (HITS)

Kleinberg [1999] proposes an approach to compute two scores for each vertex within a graph. A *hub score* assigns high ratings to vertexes with many outgoing edges. An *authority score* assigns high ratings to vertexes with many incoming edges. Originally, this algorithm was applied to rate relevancies of hyperlinked Web pages. The intended idea behind HITS is that relevant hubs link to relevant authorities and vice versa.

Definition 5.17 (*Authority, Hub*)

Based on an adjacency matrix A in which a value of one denotes an existing link and its transposed version A^T, the equations of authority and hub values a, h can be written as:

$$a^{(t+1)} = A^T h^{(t)} = (A^T A) a^{(t)}$$
$$h^{(t+1)} = A a^{(t)} = (A A^T) h^{(t)}$$

Here, t represents one step along a graph traversal. Finally, hub and authority values have to be computed asymptotically by $h = lim_{t\to\infty} h^t$ and $a = lim_{t\to\infty} a^t$. Following Ng et al. [2001] and Langville et al. [2009], a solution of these equations corresponds to the first principle eigenvector of $(A^T A)$ in terms of authority and to the first principle eigenvector of (AA^T) in terms of hub ratings.

PageRank

Page et al. [1999] propose a calculation of an authoritativeness based on an infinite random walk on a *probabilistic transition matrix*. The underlying idea is that each time a random Web surfer visits a Web page, he randomly selects from the outgoing hyperlinks the next page to visit. PageRank contains a teleporting functionality by which with a probability of ϵ the random surfer decides to jump to a Web page picked uniformly and at random from the collection of Web pages. Ng et al. [2001] explained that if teleporting is represented by a matrix U, which is uniformly populated with the value $1/n$, pagerank scores can be computed by multiplying $(\epsilon U + (1-\epsilon)M)^T$ with its principle eigenvector p.

Definition 5.18 (*Pagerank*)

$$pagerank = (\epsilon U + (1 - \epsilon)M)^T p$$

5.4.7 Maximum entropy models

Maximum entropy classifiers are widely used for a variety of natural language processing tasks, such as POS-tagging, or text segmentation [Jurafsky and Martin, 2008]. Nigam et al. [1999a] describe maximum entropy as technique for estimating probability distributions from sample data.

A maximum entropy classifier is used to predict the classes of recognized entities in text. Again, instances are represented by features in a matrix F. Here, each column represents a feature f_j that was extracted from text. Given a training corpus, in which entities were labeled with classes of an ontology, the maximum entropy approach estimates the empirical expected value for each feature $\bar{E}(f_j)$ in terms of observed feature values $f_j(x, y)$ of instances X and labels Y in the training data Z. Hence, Z contains all pairs of co-occurring labels Y and instances X in the training corpus.

Definition 5.19 (*Empirical expectation value*)

$$\bar{E}(f_j) = \frac{1}{m} \sum_{(x,y) \in Z} f_j(x, y)$$

These expected values for observed features and labels, in addition to probabilistic assumptions that $p(y|x) \geq 0$ and $\sum_{y \in Y} p(y|x) = 1$, form the constraints for the resulting probabilistic maximum entropy distribution.

Here, *entropy* can be understood likewise to the *uncertainty* about the degree of information of a hypothesis. Jaynes [1957] defines the principle of maximum entropy that whenever nothing is known the estimated probability distribution should be as uniform as possible. Under this assumption, the best probability distribution is the one, which maximizes the entropy given the constraints from the training data. The underlying conditional entropy $H(y, x)$ is defined as:

Definition 5.20 (*Conditional entropy*)

$$H(y|x) = - \sum_{(x,y) \in Z} p(y, x) log p(y|x)$$

The model, maximizing $p(y|x)$ in the space of all possible models P, is defined as:

$$p^*(y|x) = \text{argmax}(H(y|x)_{p(y,x) \in P})$$

By using optimization techniques, such as *Lagrange multipliers* λ_i, the upper form can be derived to its upper extrema by still restricting it with $\bar{E}(f_j)$ as well as the probabilistic assumptions. Finally, $p(y|x)$ can be expressed as:

$$p(y|x) = \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x,y))}{\sum_{y \in Y} \exp(\sum_{i=1}^{n} \lambda_i f_i(x,y))}$$

Nigam et al. [1999a] give a very concise example in which the principle of maximum entropy is applied to classify texts:

Example 5.6 (*Maximum entropy*)

Consider a four-way text-classification task where we are told only that on average 40% of documents with the word "professor" in them are in the faculty class. Intuitively, when given a document with "professor" in it, we would say it has a 40% chance of being a faculty document, and a 20% chance for each of the other three classes. If a document does not have "professor" we would guess the uniform class distribution, 25% each. This model is exactly the maximum entropy model that conforms to our known constraints. [Nigam et al., 1999a]

Due to probabilistic nature of the returned model distribution, the maximum entropy classifier is referred to as probabilistic classifier. This property is useful as it presumes that for a given observation O the classifier's prediction is a ordered list L of labels $L_i \in L$ along probabilities that sum up to one ($0.0 \geq p(l) \geq 1.0$ and $\sum_{i=1}^{n} p(L_i) = 1.0$). It allows the application of certainty tests that check the difference between the two highest probabilities.

Definition 5.21 (*Certainty*)

$$certainty(L) = p(L_1) - p(L_2)$$

The higher the value of *certainty* is, the more certain is the classifier about its prediction. In consequence, a threshold checks, if the certainty of a prediction is above a given value t. Compared to other classifiers such as *Naive Bayes* [Duda et al., 2001], the guarantee of probability assumptions offered by the maximum entropy model is a feature that led to the decision of its usage.

5.4.8 Conditional Random Field

CRF-models are applied to tag a sequence of words with labels that determine each word to be part of a phrase. CRF are a popular state-of-the-art approach for learning a probabilistic distribution of transitions within sequences. The following explanation of CRF will remain at a higher abstraction level. The mathematical background of a CRF

depends on mathematical frameworks, for which an explanation is out of scope of this work.

In general, CRFs apply the idea of maximum entropy models and transfer it from relational classification to sequential classification by modeling conditional dependencies between sequences and features as factor graphs [Kschischang et al., 2001]. Being trained on a corpora of sequences of words and labels, for an unlabeled sequence of words the CRF estimates the most probable distribution of labels by maximizing the entropy based on best fitting cliques within the factor graphs.

Therefore, the CRF remains a probabilistic classifier. Due to its sequential nature, it may be referred to as a probabilistic finite state transducer [Lafferty et al., 2001] as described in Section 2.2.2. For a deeper overview on CRFs, please refer to Sutton and McCallum [2006].

5.5 Summary and Conclusion

Contributing to the Hypothesis H.1, this chapter illustrated the concept of linking parts of RDF graphs to named entities in text. For this purpose, the notions of semantic links and semantic entities are defined (see Definition 5.1). Finally, this involves the specification of the Semantic Entity Recognition process (see Section 5.3).

5.5.1 Summary

This chapter described formal aspects of utilizing information from RDF graphs in information extractors:

1. Starting with Section 5.1 foundational approaches were presented providing background for linking information represented in RDF and natural language text.

2. Section 5.2 addressed the value of RDF-components for IE-tasks. It reveals the high potential of enhancing IE-systems by enriching them with existing RDF data.

3. In Section 5.3, traditional NER was extended and refined to handle semantic links and semantic entities. Finally, the Semantic Entity Recognition process allows the initial motivation of extending IE with RDF to take concrete shape.

4. For incorporating RDF knowledge into the Semantic Entity Recognition process, Section 5.4 illustrated mathematical representation and processing techniques. These techniques provide pre-processing (see Chapter 6) and processing approaches (see Chapter 7) with a mathematical framework for allowing a numeric computation on RDF represented information. Hereby, the used representational techniques are independent from the instantiated information in RDF graphs. This supports the adaptivity claimed in the Hypothesis H.3.

5.5.2 Conclusion

The presented utilization of RDF graphs within information extractors results in contributing the following solutions listed in Section 1.3:

Contribution 1 addresses the rehashing of information in RDF to compute with it in information extractors. Part of this contribution is the creation of semantic links between RDF graphs and text, and in consequence, the recognition of semantic entities as parts of RDF graphs (see Definition 5.1). The algebraic matrix representation of RDF graphs in Section 5.4.2 provided a basis for a computational interpretation and consumption of contained information.

Contribution 2 covers the Semantic Entity Recognition process, which was described and specified in Section 5.3 (see Figure 5.3).

Contribution 3 focusses on the value of using the formal vocabularies in RDF graphs. Section 5.2 exposed components within RDF graphs that are support the Semantic Entity Recognition process.

6 Preprocessing feature descriptions from text and RDF graphs

> An ideal feature extractor would yield a representation that makes the job of the classifier trivial; ...
>
> *(Duda et al. [2001])*

Information extractors, each processing different extraction tasks, require specialized bits of background information. For instance, the recognition of named entity references requires information about proper names in the text. The disambiguation of entities can be fostered with knowledge about the relationships between instance referents. Following Duda et al. [2001], extracting and selecting features from RDF graphs and text corpora is required and facilitates implementing the intended information extractors.

This chapter outlines a number of feature models that capture individual aspects of knowledge from text and RDF graphs. First, Section 6.2 demonstrates a specialized storage of RDF graphs to be used by information extractors. Next, Section 6.3 presents approaches summarizing correlating classes in taxonomies. Sections 6.4 and 6.5 illustrate statistical analyzes of knowledge in RDF graphs and text corpora. A statistical model that fosters the recognition of proper names in text and RDF is presented in Section 6.6. Next, Section 6.7 demonstrates a technique that learns how to align formal syntax descriptions about entities to datatype properties of the RDF graph. Subsequently, Section 6.8 describes the automatic creation of training data for classifying entities. Section 6.9 summarizes experiments supporting the value of presented approaches. Finally, Section 6.10 concludes valuable insights of these approaches that foster IE processors.

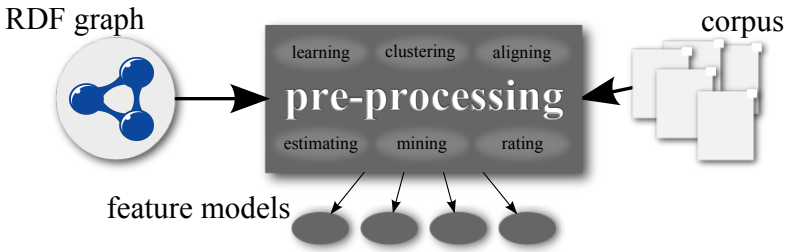


Figure 6.1: Model creation by pre-processing RDF and corpus data.

6.1 Features in Information Extraction

In general, information extractors involve various kinds of background knowledge, i.e., statistics about term frequencies, grammar-based language models, or formal synonym relationships. In this work, a special form of representing knowledge is used to enhance information extractors, namely RDF graphs. Figure 6.1 illustrates the use of pre-processors for extracting, selecting, and representing bits of knowledge as *features* from RDF graphs and text corpora. The following techniques for pre-processing RDF graphs are proposed in this chapter:

1. A relational model of RDF data (see Section 6.2), which provides the basis for handling RDF data independently from its describing domain of concern. The intention of the approach originates from the hypotheses H.1 and H.3. Its implementation supports the contributions C.1 and C.6 directly, C.2 indirectly.

2. Clustering correlating classes in RDF graphs (see Section 6.3) reduces information theoretic complexities within RDF graphs, which is required by the Hypothesis H.1 for utilizing knowledge from RDF graphs efficiently. Results of the investigated clustering algorithms support the Contribution C.1.

3. RDF graph statistics (see Section 6.4) describe the distributions of intrinsic features within RDF graphs for utilizing knowledge from RDF graphs efficiently. This corresponds with the Hypothesis H.1 and facilitates contributing C.1.

4. Text corpus statistics (see Section 6.5) enrich the descriptive statistics of RDF graph based features with descriptions in natural language text corpora. Following the line of the Hypothesis H.1 the presented maps information of an RDF graph to correlating statistical features of a text corpus, which creates a positive impact on Contribution C.1.

5. Mining datatype properties for proper names (see Section 6.6) allows rating datatype properties of an RDF vocabulary, which is in scope of Contribution C.1. With re-

spect to the hypotheses H.1 and H.3, such ratings are the basis for filtering and ranking extraction relevant vocabulary tokens needed to create the claimed adaption to individual domains of concerns by the Contribution C.6 .

6. Aligning datatype properties with regular expressions (see Section 6.7) enriches the IE process as claimed by the Hypothesis H.1. Such a utilization of regular expressions in terms of RDF graphs extends the traditional NER approach, which is conforming to Contribution C.2.

7. Automatically labeling a text corpus with classes (see Section 6.8) is required for adapting an IE system to RDF descriptions of arbitrary domains of concerns like it is claimed by the Hypothesis H.3. The presented approach as such is in line with the described Contribution C.5.

6.2 A relational model of RDF data

When providing parts of knowledge from RDF graphs to information extractors, the first problem arising is to store RDF encoded knowledge in a way that enables efficient access to information extractors. The design of RDF intends to provide arbitrary applications with a data interoperability layer on the Web. By nature, each application handles its data differently, which involves implementing specialized strategies to store and request required data. The following list summarizes information demands and data operations of the IE-algorithms that were developed and used within this work:

Requirements

1. The recognition of semantic entities involves string-based comparisons between RDF literals and text segments. Hence, requests for sorted lists of literal values from the RDF graph must be computed in minimal amount of time (as it will be described in Section 7.3).

2. The application of suffix arrays requires the computation of prefix values of a fixed length for each literal value. (This will be explained in Sections 5.4.1, 7.3).

3. The syntax analysis of datatype property values (that is going to be explained in Section 6.7) requires large lists of literals to be matched with regular expressions.

4. The disambiguation of entities' instances referents (more details will be provided in Section 7.5) as well as investigations on formal relations between recognized instances (which will be explained in Sections 7.6, 7.8) involve the application of link traversals in RDF graphs. Some kinds of links like `rdf:type` predicates are of greater focus (more details will follow in Sections 6.3, 6.4, 6.8, and 7.7). Hence, specialized indexing mechanisms should be provided.

5. Statistics on various kinds of distributions about datatype and object property values will be computed (as described in Sections 6.4, 6.6). Hence, analytical processing methods, such as selection and grouping of data, have to be provided separately for datatype and object properties.

Concept

In order to cope with these requirements, RDF triples were separated into two categories:

Definition 6.1 (*Categories of RDF triples*)

Assuming RDF triples in RDF graphs to consist of a subject, a predicate, and an object, the following distinction can be made:

Symbols *determine RDF triples that possess* datatype properties *as predicates. In consequence, all symbolic triples possess literal values as object. (e.g.,*

`dbp:Horst_Zuse rdfs:label ''Horst Zuse'' .`

)

Relations *determine RDF triples that possess* object properties *as predicates. By nature, these relational triples possess URI references or blank nodes as object values. (e.g.,*

`dbp:Horst_Zuse foaf:knows dbp:Konrad_Zuse .`

)

For storing RDF in a way that supports the IE-requirements, relational databases were considered as grounding storage technology. Relational databases provide scalable functionalities for storing, indexing, and sorting data.

In terms of RDF, several approaches exist that handle RDF graphs in relational database schemes. Currently, Jena [Wilkinson et al., 2003], Sesame [Broekstra et al., 2002], and Virtuoso [OpenLink Software Documentation Team, 2011] are the most popular RDF repositories. They all store RDF triples and process SPARQL queries. As back-end, all approaches provide the use of a relational database.

In terms of a relational database, the used schema, in general, provides a single table holding subject, predicate, and object values of RDF triples. In order to save memory and to support efficient comparisons, dictionaries are used to substitute URI values and literal values with integer identifiers. Unfortunately, neither Jena, Sesame, nor Virtuoso provides any functionality to generate prefix strings from literal values.

The approach of a single database table for representing RDF triples, as used by Jena, Sesame, and Virtuoso, is not confirm with the conceptual separation between symbols and relations. Hence, the following relational database schema was developed

and populated with RDF data. It enables the use of Structured Query Language (SQL) to query parts of the RDF graph.

Query 6.1 (*Relational schema for representing symbols and relations*)

```
-- A dictionary for indexing literal values.
CREATE TABLE index_literals (
  index SERIAL PRIMARY KEY,     -- internal key for literals
  literal varchar(256),         -- plain literal value
  prefix int );                 -- hash value for prefixes

 -- A dictionary for indexing URIs that represent instances.
CREATE TABLE index_resources (
  index SERIAL PRIMARY KEY,     -- internal key for URIs
  uri varchar(256) UNIQUE);     -- plain URI representation

 -- RDF triples, which assign literal values to instances.
CREATE TABLE symbols (
  subject int   REFERENCES index_resources(index),
  predicate int REFERENCES index_resources(index),
  object int    REFERENCES index_literals(index));

-- RDF triples, which associate two instances.
CREATE TABLE relations (
  subject int   REFERENCES index_resources(index),
  predicate int REFERENCES index_resources(index),
  object int    REFERENCES index_resources(index));
```

The dictionaries `index_resources` and `index_literals` substitute literals and URIs with numerical values. The tables `symbols` and `relations` represent the conceptual separation of RDF triples.

To further support prefix-based substring comparisons performed by suffix arrays, each lexical prefix of a literal is indexed and hashed. Hashing prefixes of literals allows requesting a list of literal values that possibly match with a passed prefix string of a suffix array entry. Prefixes of RDF literals are hashed with numeric values by using the following hash function, which interprets a string as array of characters.

Definition 6.2 (*Hashing string values*)

This hash function is used by the Java programming language for computing hashes of string objects.

$$hash(string) = \sum_{i=0}^{n=|string|} string[i] * 31^{n-(i+1)}$$

In Section 7.3, the length of hashed string prefixes will be parameterized. Furthermore, results of experiments (that will be described Section 7.9.3) focus on the impact of differing lengths (|string| = 1,2,3,4, or 5) of hashed string prefixes on computing complexity.

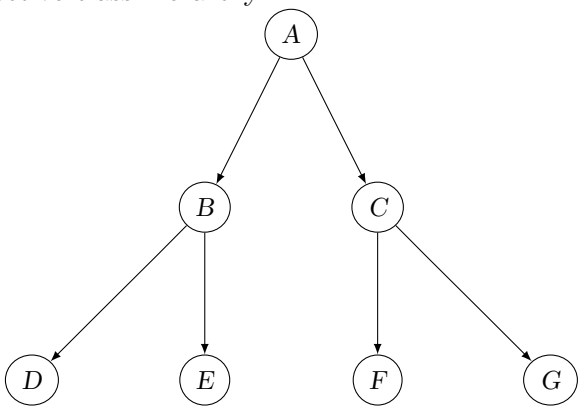
In this work, the relational database `PostgresQL`[1] was used as basic storage implementation. In practice, the schema in Listing 6.1 was populated with hundreds of millions of RDF triples, such as provided by the DBpedia dataset. Still, the database on this schema is capable to return query results quickly, even when sending complex queries with multiple join operations.

6.3 Clustering correlating classes in RDF graphs

Instances within RDF graphs are classified along class hierarchies (also referred to as taxonomies). Figure 6.1 illustrates an example class hierarchy consisting of a root node A with two nested descendants B and C.

Example 6.1 (*Example class hierarchy.*)

Assuming the nodes A,B,C,D,E,F,G to be classes in an RDF graph. The following tree illustrates their respective class hierarchy.



In a class hierarchy of A, B, C, the two sibling classes B and C, which are subsumed by a third class A, indicate that B and C share a specific nature, which is summarized by A. In RDF, such a shared nature can be explicated with datatype and object properties by restricting their signatures (i.e., domain, range) to A. Conversely, the specific natures of classes B and C can be defined by modeling specialized properties with signatures restricting domain or range to B or C. In the absence of such discriminating properties, it is hard to choose correct classes for instances without knowing explicit classifying relations such as `rdf:type`. related to this, in the area of IE it can not be asserted that the surrounding text of a recognized entity holds enough evidence for discriminating

[1] `http://www.postgresql.org/`

between A, B, and C. In this work, we refer to this problem as *over-specification of instance referents in RDF graphs*. Examples 6.2 and 6.3 illustrate this problem:

Example 6.2 (*RDF properties subsuming or specializing classes*)
Assuming the following taxonomy in RDFS:

```
dbp-ont:Athlete      rdfs:subClassOf      foaf:Person      .

dbp-ont:Politician     rdfs:subClassOf     foaf:Person      .
```

The datatype properties `foaf:firstName` and `foaf:lastName` define person names. Hence, their domain is defined on instances of type `foaf:Person`. In consequence, instances of the sibling classes B, C (here, `dbp-ont:Athlete`, `dbp-ont:Politician`) can both be described by using these properties.

Conversely, the object property `dbp-ont:club` associates a `dbp-ont:Athlete` with a sports club, whereas `dbp-ont:party` defines instances of `dbp-ont:Politician` as members of political parties.

Example 6.3 (*Over-specification of instance referents in RDF graphs*)
Assuming the following sentence:

> *"Christian Wulff presents Manuel Neuer, the captain of Schalke's soccer team, with the DFB cup."*

It can be inferred that "Manuel Neuer" and "Christian Wulff" are both names of persons. Furthermore, it can be inferred that the person Manuel Neuer is a soccer player and a member of a sports team, which classifies him as athlete. The text does not provide any information that, for example, refers to the politician nature of Christian Wulff. (In 2010, Christian Wulff was elected as President of Germany)

Features extracted from natural language text do not always possess enough information to determine a correct classification of entities in terms of a deeply nested taxonomy. This issue is going to cause problems when training, for example, an entity classifier. In general, whenever the two classes B and C can hardly be discriminated, by implication, a correlation between B and C can be assumed. Hence, the goal is to cluster those correlating classes that can hardly be discriminated. The following algorithm extracts correlating coefficients between all classes by sampling instances from RDF graphs:

Algorithm 6.1 (*Mine correlating classes by co-occurrence*)

Input: *An RDF graph G consisting of a set of instances I, which are classified to a set of classes C; a parameter m, which denotes the amount of sample instances, which should be analyzed from G.*

Output: *A correlation matrix M_{COR}.*

1. *Create the $n \times n$ matrix M_{COOC}. n denotes the number of distinct classes in G.*
2. *Extract m random sample instances $I_m^c \subset I$ for each class $c \in C$ from G.*
3. *For each instance sample $i \in I_m^c$, retrieve its associated set of classes $C_i \subset C$.*
4. *For each pair of classes $c_i \in C_i$, $c_j \in C_i$ increase $M_{COOC}(i, j)$ by one.*
5. *The resulting matrix M_{COOC} is referred to as **co-occurrence matrix**. The columns j of a row $M_{COOC}(i)$ describe how often instances of type c_i are also classified as c_j.*
6. *By using Pearson's correlation coefficient (see Section 5.4.4), transform the co-occurrence matrix M_{COOC} into a **correlation matrix** M_{COR}.*

Based on the taxonomy, Figure 6.4 illustrates the derived co-occurrence matrix M_{COOC}, which can be sampled by using the algorithm described above. Here, only a single instance per class was assumed. The co-occurrence values in M_{COOC} were transformed into the correlation matrix M_{COR}, which is shown by Figure 6.5.

Example 6.4 (*Co-occurrence matrix (M_{COOC})*)

The co-occurrence matrix (M_{COOC}) is derived from the taxonomy in Example 6.1.

$$M_{COOC} = \begin{pmatrix} & A & B & C & D & E & F & G \\ A & 7 & 3 & 3 & 1 & 1 & 1 & 1 \\ B & 3 & 3 & 0 & 1 & 1 & 0 & 0 \\ C & 3 & 0 & 3 & 0 & 0 & 1 & 1 \\ D & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ E & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ F & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ G & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Example 6.5 (*Correlation matrix (M_{COR})*)

The correlation matrix (M_{COR}) is derived from Example 6.4.

$$M_{COR} = \begin{pmatrix} & A & B & C & D & E & F & G \\ A & \mathbf{1.0} & 0.7 & 0.7 & 0.5 & 0.5 & 0.5 & 0.5 \\ B & 0.7 & \mathbf{1.0} & -0.0 & 0.8 & 0.8 & -0.1 & -0.1 \\ C & 0.7 & -0.0 & \mathbf{1.0} & -0.1 & -0.1 & 0.8 & 0.8 \\ D & 0.5 & 0.8 & -0.1 & \mathbf{1.0} & 0.4 & -0.2 & -0.2 \\ E & 0.5 & 0.8 & -0.1 & 0.4 & \mathbf{1.0} & -0.2 & -0.2 \\ F & 0.5 & -0.1 & 0.8 & -0.2 & -0.2 & \mathbf{1.0} & 0.4 \\ G & 0.5 & -0.1 & 0.8 & -0.2 & -0.2 & 0.4 & \mathbf{1.0} \end{pmatrix}$$

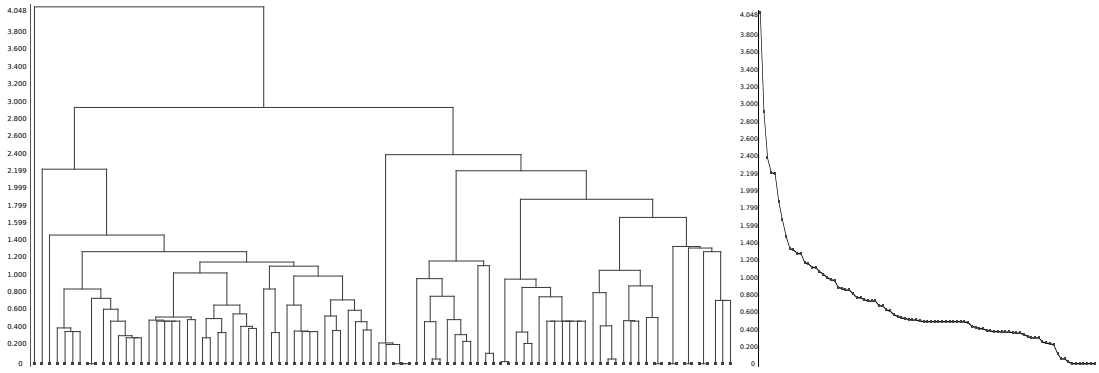
6.3.1 Hierarchical clustering

The hierarchical clustering algorithm, which is described in Section 5.4.3, allows the clustering of correlating classes in a matrix M_{COR}. The *Euclidean distance* (see Section 5.4.2) is used for computing the similarity between two classes. Similarities between two clusters are calculated by taking the mean distance of all contained classes.

The dendrogram in Figure 6.2a illustrates a tree of clusters along a one-dimensional scale of accumulated euclidean distances. Besides the dendrogram, the curve in Figure 6.2b describes the decrease of mean Euclidean distance ratios between the closest clusters, when reducing the amount of clusters.

Finally, for retrieving a fixed set of clustered RDF classes, it is necessary to provide a threshold value t_k of mean Euclidean distances as upper bound. It determines that all k clusters with Euclidean distance below t_k have to be returned as significant clusters. As the hierarchies of classes and therefore the correlations between these classes differ between RDF graphs, the optimal threshold t_k has to be evaluated for each RDF graph. Therefore, Sugar and James [2003] describe an approach to automatically estimating a good number of clusters, which may provide a further automatism in future work.

Labeling clusters of correlating classes For each cluster passing the threshold test, a *representative class* has to be identified, which subsumes the contained classes within the cluster. For this reason, the co-occurrence counters in M_{COOC} are transformed into conditional probabilities in a matrix M_{COPR}:



(a) (b)

Figure 6.2: (a) Clustering correlating classes along an y-axis of distance values. (b) Regression of distance values by increasing count of clusters.

Definition 6.3 (*Conditional probability matrix*)

Based on co-occurrences within the transitive closure of classes associated to instances, the following conditional probabilities can be computed expressing how likely an instance of class c_i is also classified with c_j.

$$M_{COPR}(i,j) = \qquad P(C_i|C_j) = \qquad \frac{|C_i \bigcap C_j|}{|C_j|}$$

The conditional probabilities $P(c_i \in C|c_j \in C)$ denote how likely an instance of type c_j is also of type c_i. Hence, if c_j is subclass of c_i, it can be assumed that $P(c_i|c_j) > P(c_j|c_i)$. Consequently, the upper right triangle of M_{COPR} contains greater values than the lower left triangle. The interpretation of values in the upper right triangle ($i > j$) allows to infer for all $M_{COPR}(i,j) > 0$ that c_i is a super class of c_j. The lower left triangle ($i < j$) denotes a taxonomic distance between c_i as being a super class of c_j. The higher the value is, the smaller is the distance. The conditional probability matrix M_{COPR} can be interpreted as adjacency matrix (see Section 5.4.6) that represents the transitive closure of a hierarchy as directed graph with weighted edges. Example 6.6 presents an extreme variant of M_{COPR} in which the upper right triangle contains values either of zero or of one.

Example 6.6 (*Conditional probability matrix M_{COPR}.*)

Conditional probability matrix M_{COPR}, derived from data of Example 6.4. $P(c_i|c_j)$ denote how likely an instance of type c_j is also of type c_i.

$$M_{COPR} = \begin{pmatrix} & A & B & C & D & E & F & G \\ A & \mathbf{1.0} & \mathbf{1.0} & \mathbf{1.0} & \mathbf{1.0} & \mathbf{1.0} & \mathbf{1.0} & \mathbf{1.0} \\ B & 0.3 & \mathbf{1.0} & 0.0 & \mathbf{1.0} & \mathbf{1.0} & 0.0 & 0.0 \\ C & 0.3 & 0.0 & \mathbf{1.0} & 0.0 & 0.0 & \mathbf{1.0} & \mathbf{1.0} \\ D & 0.1 & 0.3 & 0.0 & \mathbf{1.0} & 0.0 & 0.0 & 0.0 \\ E & 0.1 & 0.3 & 0.0 & 0.0 & \mathbf{1.0} & 0.0 & 0.0 \\ F & 0.1 & 0.0 & 0.3 & 0.0 & 0.0 & \mathbf{1.0} & 0.0 \\ G & 0.1 & 0.0 & 0.3 & 0.0 & 0.0 & 0.0 & \mathbf{1.0} \end{pmatrix}$$

The calculation of a cluster's representative is based on building two sums of conditional probabilities of:

- $c_i \in C$ for each pair of $P(c_i|c_j)$ called ancestors *anc* and
- $c_j \in C$ for each pair of $P(c_j|c_i)$ called distances *dist*.

Both sums are then divided by the number of clusters $|C^{\text{cluster}}|$ in order to assure values between zero and one:

Definition 6.4 (*Ancestor ratios in taxonomies*)

Related to the upper right triangle of M_{COPR}, the following sum aggregates probabilities of c_i being a super class.

$$anc(c_i \in C) = \frac{\sum_{j \in M_{COPR}} P(c_i \in C | c_j \in C^{cluster})}{|C^{cluster}|}$$

Definition 6.5 (*Distance ratios in taxonomies*)

Related to the lower left triangle of M_{COPR}, the following sum aggregates the taxonomic distance between c_i and the remainder classes.

$$dist(c_i \in C) = \frac{\sum_{j \in M_{COPR}} P(c_j \in C | c_i \in C^{cluster})}{|C^{cluster}|}$$

Algorithm 6.2 (*Labeling clusters*)

Input: *A set of classes C; a clustering of these classes; c_i, which determines the set of classes in the RDF graph; c_j, which determines the set of classes in the cluster.*

Output: *The representative class for each cluster of correlating classes.*

1. *For each $c_i \in M_{COPR}$ calculate its ancestor values $anc(c_i)$.*
2. *For each $c_i \in M_{COPR}$ calculate its distance values $dist(c_i)$.*
3. *Return the class that possesses the highest product value between $dist$ and anc is taken as label. $(c_i | \max_{c_i}\{dist(c_i) \times anc(c_i)\})$*

6.3.2 Principle Component Analysis

As alternative to hierarchical clustering, the PCA, which is described in Section 5.4.5, is applied to the correlation matrix M_{COR}. Again, the goal is to transform class representations i in M_{COR} into a lower dimensional space $n_r < n$. Depending on a given proportion $0.0 < t < 1.0$, the dimensional reduction of a PCA returns a matrix M_{RED}, which consists of a reduced number $n_r < n$ of columns.

Example 6.7 (*PCA-reduced matrix M_{RED}*)

The PCA-reduced matrix M_{RED} is derived from data of Example 6.4.

$$M_{RED}^{PCA} = \begin{pmatrix} & CLUSTER_1 & CLUSTER_2 & CLUSTER_3 \\ A & \mathbf{5.394} & 0.000 & 0.000 \\ B & 0\mathbf{.763} & -2.344 & 0.000 \\ C & 0.763 & \mathbf{2.344} & 0.000 \\ D & -1.73 & -0.858 & \mathbf{-0.62} \\ E & -1.73 & -0.858 & \mathbf{0.62} \\ F & -1.73 & \mathbf{0.858} & -0.34 \\ G & -1.73 & \mathbf{0.858} & 0.34 \end{pmatrix}$$

For each row, the index $0 \leq j < n_r$ of the maximum column value determines the cluster the class in row i is assigned to. Within each cluster j, the class i with the maximum value $M_{RED}^{i,j}$ is selected as label of the cluster. In Figure 6.7, the PCA was set to reduce the count of columns to three. The maximum column values of each row, which are marked as bold, can be used to cluster each instance.

6.4 RDF graph statistics

A statistical graph analysis allows recognizing recurring patterns of knowledge about instances in the RDF graph that may (if utilized) foster IE. Such an analysis comprises, for example, the recognition of statistical patterns in the distribution of datatype properties as well as object properties.

6.4.1 Usage statistics of datatype properties

In RDF, datatype properties describe literal-valued identifiers, names, or other kind of attributes of instances. For utilizing such literal values to, e.g., for enhancing the recognition of entities in text, it is necessary to know which datatype properties are more suitable than others are.

In general, for fostering IE by applying datatype properties of an RDF graph, the effective application value of each datatype property depends on:

1. A high *coverage* of instances of allowed classes (with respect to the properties' signature definitions) that assign values to these datatype properties.

2. A low amount of *ambiguity*, resulting from the number of instances that share the same datatype property values.

3. The frequent use of especially these datatype property values in text (*document frequency*).

A statistical graph analysis allows the computation of *coverage* and *ambiguity*. Section 6.5 will illustrate methods for computing the *document frequency* by using additional knowledge from text corpora.

Coverage In terms of IE, a relevant datatype property assigns literal values to a high coverage of instances of a certain class (e.g., most persons in DBpedia possess `foaf:name` values).

Definition 6.6 (*Coverage*)
For a given class t, the **coverage** *of a datatype property p is introduced and defined as the count of instances s of t that possess p as predicate in RDF triples of the RDF graph G, divided by the overall number of instances of t in G:*

$$coverage(p, t, G) = \frac{|\{s|\forall s \in t \wedge \exists (s, p, o) \in G\}|}{|t|} \tag{6.1}$$

Ambiguity In addition to high coverage, values of extraction-relevant datatype properties possess minimal ambiguities. In consequence, it is desirable to select only those datatype properties that possess literal values, which in average refer to merely a single instance.

Definition 6.7 (*Ambiguity*)

The lexical ambiguity of a literal l in an RDF graph G is introduced as the distinct number of subjects s in RDF triples that possess l as object.

$$\textit{lexical ambiguity}(l, G) = |\{s|\exists(s, p, l) \in G\}| \tag{6.2}$$

Subsequently, the average ambiguity of a datatype property p is determined by the mean value of lexical ambiguities of literal values l in existing RDF triples (s, p, l).

$$\textit{ambiguity}(p, G) = \frac{\sum_{l}^{\exists(s,p,l)\in G} \textit{lexical ambiguity}(l, G)}{|\{l|\exists(s, p, l) \in G\}|} \tag{6.3}$$

An analysis of datatype properties of the RDF graph of DBpedia revealed that on average `rdfs:label` and `foaf:name` were rated best by coverage and ambiguity (see Section 6.9.3).

In order to compute coverage and ambiguity ratios, the relational database schema presented in Section 6.2 was extended. The SQL view in Listing 6.2 creates an overview on the count of instances per class that exists within an RDF graph.

Query 6.2 (*Histogram of count of instances per class*)

```
-- Creates view on how many instances per class exist.
CREATE VIEW histogram_types AS
  SELECT r.object AS type, count(DISTINCT r.subject) AS count
  FROM relations r WHERE (r.predicate IN (
      SELECT R.index FROM index_resources R
      WHERE R.uri = 'rdf:type'))
  GROUP BY r.object;
```

In Query 6.3, the SQL view creates an overview of the count of instances that share literal values within an RDF graph.

Query 6.3 (*Histogram about the count of distinct subjects per literal value*)

```
-- Creates a view on how many instances assign equal literal values.
CREATE VIEW histogram_literals AS
  SELECT S.object AS literal, count(DISTINCT S.subject) AS count
  FROM symbols S GROUP BY S.object
```

Finally, the SQL Query 6.4 uses the histogram on literals to calculate ambiguity values for each datatype property.

Query 6.4 (*Calculating ambiguity values*)

```
-- A view, which computes ambiguity values for each datatype property.
CREATE VIEW AMBIGUITY_SYMBOLS AS
  SELECT S.predicate as property, avg(HL.count) as ambiguity
  FROM symbols S, histogram_literals HL
  WHERE ( HL.literal = S.object )
  GROUP BY symbols.predicate
```

These three views are necessary to formulate the following SQL query, which calculates for each datatype property and class the degree of coverage and ambiguity.

Query 6.5 (*Computation of coverage and ambiguity*)

```
-- Select query, which returns coverage and ambiguity values
-- for each datatype property.
SELECT A.property, C.coverage, A.ambiguity FROM (
  SELECT S.predicate AS property,
         (count(DISTINCT S.subject)/avg(HT.count)) AS coverage
  FROM relations R, symbols S, histogram_types HT
  WHERE ( HT.type = R.object AND
      S.subject = R.subject AND R.object = <ID of class> )
    GROUP BY S.predicate) C, AMBIGUITY_SYMBOLS A
WHERE (A.property = C.property)
```

An evaluation of these ratios is described in Section 6.9.3.

6.4.2 Estimating cardinalities

The cardinalities of datatype and object properties can be divided into four categories:

1 : 1 A property with a 1 : 1 cardinality associates subject and object only once. Any kind of unique identifiers such as tax IDs, social security numbers, or ISBN numbers may be examples for this kind of property.

1:m Objects must only be referred to once by a subject when using such properties. In mathematics, this cardinality is referred to as injective function. In OWL it is defined as *inverse functional property*. An example is the modeling of email addresses of abstract agents in the FOAF vocabulary.

n:1 Instances may possess properties with this cardinality only once. In mathematics, this cardinality is referred to as surjective function. OWL defines it as *functional property*. For example, in western culture the spouse relationship, and in general, father and mother relationships are defined as functional.

n:m Any subject may be related with any object by using properties of this cardinality. The friendship relation is an example for this type of cardinality.

For each datatype property, Query 6.6 calculates cardinalities of domain values.

Query 6.6 (*Calculating domain cardinalities*)

```
-- A view, which calculates the cardinalities of domain values
-- for all datatype properties.
CREATE VIEW SUBJECT_CARD_SYMBOLS AS
  SELECT H.predicate, count(distinct H.subject),
         sum(H.C), sum(H.C)/count(distinct H.subject)
  FROM ( SELECT subject, predicate, count(*) AS C FROM symbols
  GROUP BY subject, predicate) AS H
GROUP BY H.predicate
```

For each datatype property, Query 6.7 calculates cardinalities of range values.

Query 6.7 (*Calculating range cardinalities*)

```
-- A view, which calculates the cardinalities of range values
-- for all datatype properties.
CREATE VIEW OBJECT_CARD_SYMBOLS AS
  SELECT H.predicate, count(distinct H.object),
         sum(H.C), sum(H.C)/count(distinct H.object)
  FROM ( SELECT object, predicate, count(*) AS C FROM symbols
  GROUP BY object, predicate) AS H
GROUP BY H.predicate
```

By replacing table `symbols` with table `relations`, the upper two views calculate cardinalities of object properties. Knowledge about cardinalities will be used in the prediction of unknown relations between recognized instances (see Section 7.8).

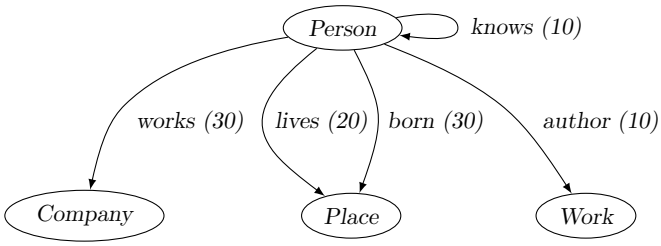
6.4.3 Estimating characteristic relations between classes

The recognition of relationships between entities in text can be fostered by incorporating the relational knowledge from RDF graphs. In RDF, object properties represent relationships between instances. Here, respective signatures define valid classes of instances. Characteristic relationships between instances of classes A and B can be computed by analyzing the distribution of used object properties between A and B.

Figure 6.8 illustrates an example, in which object properties of 50 instances of type Person are rendered as directed graph.

Example 6.8 (*Relations of persons.*)

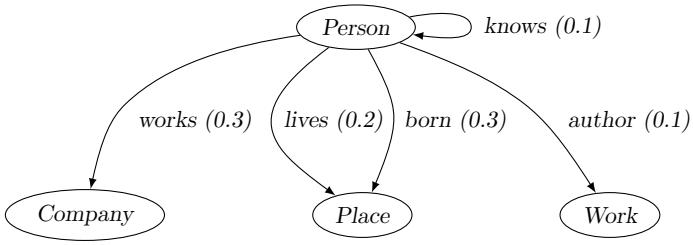
Object relations of a sample of 50 instantiated persons.



A model for representing graph-based relationships is a probabilistic transition matrix. In Figure 6.9, the absolute edge counts of the graph presented in Figure 6.8 are normalized by dividing each edge count with the overall aggregation of edge counts of outgoing edges of person. This results in a probabilistic transition graph, which can be represented as probabilistic adjacency matrix as it was already described in Section 5.4.6.

Example 6.9 (*Outgoing transitions graph starting from persons*)

This Markov chain is derived from Example 6.8.



When combining transition probabilities of characteristic relationships of each class (or cluster) of an RDF graph, the resulting transition matrix (also referred to as Markov chain) can be used as base for predicting relationships between instances. See Section 7.8 for more details about the application of Markov chains in fact prediction. Algorithm 6.3 describes how to learn a Markov chain from object properties of sampled instances.

Algorithm 6.3 (*Learning a Markov chain from object properties*)

```
def markovChain (SAMPLE_SIZE = 100) :
  MARKOV_CHAIN = {}
  CLASS_COUNT = {}
  for c in getClassesFromRDFGraph() :
        instanceList = getSampleInstances(c, SAMPLESIZE)
        for instance in instanceList :
          for predicate, object in getObjectProperties(instance) :
                CLASS_COUNT[c] += 1
                for type in getClasses(object) :
                      MARKOV_CHAIN[(c, p, type)] += 1
  for (c, p, type) in MARKOV_CHAIN:
       MARKOV_CHAIN[(c, p, type)] /= CLASS_COUNT[c]
  return MARKOV_CHAIN
```

6.5 Text corpus statistics

In general, the focus of this work is set on fostering IE by utilizing knowledge from RDF graphs. The use of statistical models derived from text corpus data is part of the state-of-the-art in IE. In the following investigation on proper nouns, a combination between text corpus statistics and RDF graph statistics is introduced. The investigation on proper nouns in RDF graphs (see Section 6.6) involves investigating the occurrence of literals in a text corpus. To start with, this analysis comprises the well-approved statistics on term and document frequencies:

6.5.1 Term and document frequencies

Manning et al. [2008] describe *term frequency* as assigning to each term in a document a weight for that term, which depends on the number of occurrences of the term in the document. In Section 7.6, term frequencies are used within relevance analyses of recognized entities.

Definition 6.8 (*Term frequency*)

Term frequency *describes the number of times n a term t occurs in document d divided by the sum of all terms in document $|d|$.*

$$tf_{t,d} = \frac{n_{t,d}}{|d|} \tag{6.4}$$

The *document frequency* of a term is defined to be the number of documents in the collection that contain the term:

Definition 6.9 (*Document frequency*)

Document frequency *of a term t describes the number n of documents D that contains the term t divided by the overall number of documents $|D|$*

$$df_{t,D} = \frac{n_{t,D}}{|D|} \tag{6.5}$$

The combination of term frequency with the inverse document frequency produces a composite weight for each term in each document:

Definition 6.10 (*tf-idf*)

tf-idf *combines term frequency (**tf**) and document frequency (**df**) by multiplying **tf** and the logarithmic inverse of **df**. The logarithm is applied to smooth resulting values. It is increased by one to avoid a division by null:*

$$tf\text{-}idf_{t,d,D} = tf_{t,d} \times log(\frac{|D|}{n_{t,D} + 1}) \tag{6.6}$$

6.5.2 Combining text corpus with RDF graph statistics

Combining the usage of term statistics in a document corpus and statistics in RDF graphs allows analyzing which datatype property values also occur as tokens in text. Here, the following metric about datatype properties describe how frequently contained values occur in text.

Definition 6.11 (*Datatype property-based document frequency*)

For given documents d in a corpus C, the document frequency of a property p is introduced as the count of those $d \in C$ which contain literal object values l of p in an RDF graph G, divided by the overall count of documents in C.

$$df(p, C) = \frac{|\{d \in C | \exists l \in d \wedge \exists (s, p, l) \in G\}|}{|C|} \tag{6.7}$$

6.6 Mining datatype properties for proper names

Datatype properties assign literal values to instances. In general, RDF graphs possess various datatype properties that describe different facets of concepts they refer to. However, not all of these properties contain the kind of literal values that can be determined

as being *proper names*. For example, values of prices, weights, and heights are not suitable for recognition of entities like books in the Amazon Book store. As many books share identical values of these properties, the recognition would result in large lists of possible books that possess at least one of these property values. Therefore, entity recognition requires *proper names* to guarantee a minimal ambiguity. The goal of this task is to mine datatype properties for those representing proper names. Here, Peirce [1976] provided a pragmatic definition on the nature of *proper names*:

Definition 6.12 (*Proper names*)

[Pietarinen, 2010, quoting Peirce] A ***proper name****, when one meets with it for the first time, is existentially connected with some percept or other equivalent individual knowledge of the individual it names. It is then, and then only, a genuine Index. The next time one meets with it, one regards it as an Icon of that Index.*

This definition refers back to Peirce's theory of signs in which an index is a sign that denotes its object by virtue of an existential connection that it has with its object. Peirce considered any designation to be an index, for example, a pronoun, a proper name, or a label on a diagram, actually directing or compelling the mind's attention toward the object just as a reagent does.[2] The nature of proper names does not assign any additional descriptions to the object they refer to (c.f.[Pietarinen, 2010]). This corresponds with the quotation by John Stuart Mill, which was introduced Chapter 5.

Following Peirce's convention of names, Semantic Web ontologies define datatype properties as links between instances and literal values. The RDF vocabulary provides the most general property for a name as `rdfs:label` which is defined as `owl:AnnotationProperty`. Values of a `owl:AnnotationProperty` only serve pure annotation purposes without any intended formal implications. Popular vocabularies model naming properties such as `foaf:name` or `dc:title` either as sub-property of `rdfs:label` or by determining these properties to be of type `owl:AnnotationProperty`.

For avoiding ambiguities while resolving entity references, the combination of *coverage*, *ambiguity*, and *inverse document frequency* assigns higher ratings to properties that are determined to contain good proper names:

Definition 6.13 (*Proper Name Rating*)

The proper name rating is introduced as follows:

$$\textit{proper name rating}(p,t,C) = \frac{\textit{coverage}(p,t)}{\textit{ambiguity}(p)} * \textit{idf}(p,C)$$

[2] As described in http://mywikibiz.com/index.php?title=Charles_Sanders_Peirce&oldid=125195#Types_of_signs; 3. June, 2011

6.7 Aligning datatype properties with regular expressions

In contrast to dictionary-based representations, regular expressions provide methods for describing the general syntax of a family of literal values such as dates, addresses, intervals, or prices. The regular expressions in Example 6.10 describe email addresses and dates.

Example 6.10 (*Regular expressions*)

Regular expressions in Python describing the syntax of email addresses and dates.

- `email = re.compile("[a-zA-Z0-9\._]+@[a-zA-Z0-9\.]+\.[a-zA-Z]+")`
- `date = re.compile("([1-9]+[0-9]+)\-([1-9]?[0-9]+)\-([1-9]?[0-9]+)")`

Based on such a syntax description, the RDF graph can be queried for datatype property values that match with these descriptions. Queries 6.8, 6.9, 6.10 and show how the relational database schema facilitates the creation of an histogram of datatype properties which aggregates values that match with a given regular expression.

Query 6.8 (*Schema extension*)

```
-- A database view, which creates a histogram on how many
-- distinct literal values exist per datatype property.

CREATE VIEW histogram_symbols AS
        SELECT predicate, count(DISTINCT object)
        FROM symbols
        GROUP BY predicate;
```

After dividing each histogram value with the cardinality of the related datatype property, datatype properties with a ratio above 0.9 are considered as relevant in terms of a given regular expression.

Query 6.9 (*Matching regular expressions with datatype property values*)

```
-- A table that holds information about properties with values
-- that match with a regular expression.
CREATE TABLE literals_regex_distribution (
  regex varchar(100), -- the regular expression
  property int,       -- the datatype property
  ratio float);       -- the ratio of matching values
```

Query 6.10 (*Matching regular expressions with datatype property values*)

```
-- Populates this table with the amount of matches
-- between values of a datatype property and a regular
-- expression. A threshold t = 0.9 assures that only those
-- properties with high amount of matches populate the table.
INSERT INTO literals_regex_distribution VALUES (
  SELECT H.P, H.C*1.0 / count FROM (
        SELECT predicate AS P, count(DISTINCT object) AS C
          FROM index_literals, symbols
          WHERE literal ~ 'regex' AND index = symbols.object
          GROUP BY predicate) AS H, histogram_symbols
        WHERE H.P = predicate AND H.C*1.0 / count > 0.9 )
```

6.8 Automatically labeling a text corpus with classes

For classifying proper nouns in text with classes from the RDF graph, it is necessary to create training data on top of which statistical distributions can be created. These statistics provide a sufficient statistical understanding for training a classifier.

Based on the clustering of correlating classes and parts of the processing steps, classes of recognized semantic entities are used as labels for the originating sequences of words, respectively. The resulting labeled text corpus can be used either as training or validation data. This requires the following list of processing tasks of the Semantic Entity Recognition process, which will be further investigated in the next chapter:

1. Filtering text for proper names (see Section 7.2)
2. Spotting text for datatype property values (see Section 7.3)
3. Linking named entities to formal instances (see Section 7.4)
4. Resolving ambiguous semantic entities (see Section 7.5)

6.9 Experiments and evaluations

The following sections present results of experiments that were performed to confirm the value of the presented pre-processing approaches.

6.9.1 Datasets

Most experiments were settled on RDF data provided by DBpedia[3] and text data from Reuters News Corpus. These datasets provide a large amount of data and cover a wide

[3]More specifically, the experiments were performed on DBpedia data of version 3.51

range of topics. Hence, the experimental results generated on these datasets contained most meaningful insights.

DBpedia DBpedia is an open source knowledge base. It provides an RDF graph about knowledge extracted from the Wikipedia [Bizer et al., 2009b] in regular cycles. Part of DBpedia's RDF graph is the DBpedia ontology, which classifies instances within the graph along a taxonomy of about 274 classes. It models several hundreds of datatype and object properties about these classes. Concrete reasons for using DBpedia data in these experiments are:

- DBpedia provides massive amount of RDF data. It consists of 14 933 856 distinct URI references, 8 019 745 instances, 9 363 625 distinct literal values, 142 559 573 RDF triples of object properties, and 13 741 397 RDF triples of datatype properties.
- DBpedia is extracted automatically from tabular-like info boxes of the Wikipedia. Hence, it contains huge set of errors. Although this fact bears several implementation problems, it ensures in general, that the developed pre-processing and extraction methods had to cope with these errors. It can be assumed that data from other real environments also contains a huge set of similar errors.
- RDF data from DBpedia is linked to natural language text in Wikipedia. This facilitates the generation of language models as done in Section 6.8.
- The domain of concern of DBpedia covers an open world scenario, which allows its interpretation even for non-experts.
- Underlying the ontology, DBpedia provides of a large amount of instances, which cover each class of the taxonomy. This allows the computation of significant statistics.
- DBpedia reused datatype and object properties from arbitrary Semantic Web vocabularies such as FOAF, Dublin Core, and GEO (see Section 1.5). This supports a realistic applicability of presented approaches, as these vocabularies are of most popular use in the Semantic Web across a variety of domains.

Reuters News Corpus Reuters is a global news agency. The Reuters Corpus, Volume 1 is a corpus of Reuters News stories for use in research and development of natural language-processing, information-retrieval or machine learning systems.[4]

In this work, experiments on text data were mainly conducted on Reuters News stories. The concrete reasons for using the Reuters Corpus are:

[4] http://trec.nist.gov/data/reuters/reuters.html

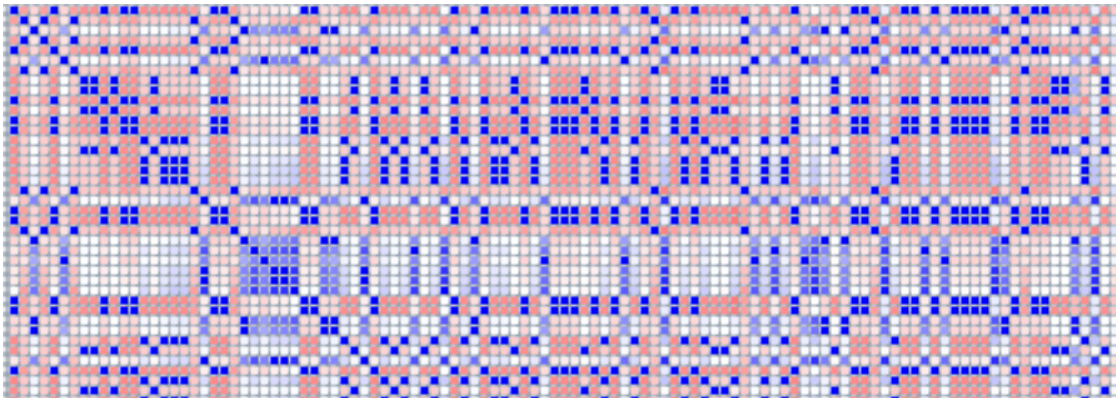


Figure 6.3: Colored extract of a correlation matrix of co-occurring classes in DBpedia. Degrees of blue represent positive and degrees of red negative correlations.

- News stories from Reuters cover a wide range of topics, including science, politics, sports, or pop gossip.
- The corpus consists of a large number of documents.
- Entity types like locations, persons, and organizations occur frequently in text.
- Entities of these types also occur in the DBpedia dataset. Table 7.7 lists concrete numbers of correlating types of entities and instances.
- The ConLL2003 dataset [Tjong Kim Sang and De Meulder, 2003] labels text of the Reuters corpus with the entity types person, place, organization, and miscellaneous.

6.9.2 Clustering classes in DBpedia

In this study, the 274 classes of DBpedia were clustered by analyzing co-occurring classes in `rdf:type` statements. Therefore, for each class a sample of 20 instances was chosen randomly. The transitive closure of classes (`rdf:type` statements) of these instances was used to populate the co-occurrence matrix M_{COOC}. The correlation matrix in Figure 6.3 was derived from such a sampled co-occurrence matrix. Figure 6.3 illustrates a colored part of the complete correlation matrix. Matrix cells are colored by degree of correlation values between minus one and one. Degrees of blue represent positive correlations. Degrees of red represent negative correlations. It can be observed that the matrix consists of rectangles of correlating regions of blue values. These correlating regions may be subsumed by using clustering approaches.

As described in Section 6.3, PCA (see Section 5.4.5) and hierarchical clustering (see Section 5.4.3) were applied to cluster correlating classes. In order to perform a quantitative rating of the quality of clusters, the metrics **taxonomic precision** (TP) and **taxonomic recall** (TR), which were coined by Dellschaft and Staab [2008], were applied. The intention of applying these metrics to evaluate the computed clusters is to compare the topology of computed clusters and their representative classes with the originating topology of the underlying taxonomy.

Definition 6.14 (*Global taxonomic precision and recall*)
Global definition of precision (TP) and recall (TR) between two taxonomies:

$$TP(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} tp(c, O_C, O_R)$$
$$TR(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} tp(c, O_C, O_R)$$

The global definition of precision and recall between two taxonomies relies on strategies that estimate local precision and recall values in terms of a single concept.

Definition 6.15 (*Local taxonomic precision and recall*)
Local taxonomic precision (tp) and recall (tr) between a concept in two taxonomies:

$$tp(c, O_C, O_R) := \frac{|ce(c, O_C) ce(c, O_R)|}{|ce(c, O_C)|}$$
$$tr(c, O_C, O_R) := \frac{|ce(c, O_C) ce(c, O_R)|}{|ce(c, O_R)|}$$

According to Dellschaft and Staab [2008], the term *ce* represents a characteristic environment of a concept within a taxonomy, which can be defined as follows:

Definition 6.16 (*Characteristic environment*)
Characteristic environment (ce) of a concept, i.e., all its sub and super concepts:

$$ce(c, O) := \{c_i | c_i \in C_O \wedge (C_i \leq c \vee c \leq c_i)\}$$

The distribution of taxonomic precision values, which is presented in Figure 6.4 shows that independent from the amount of calculated clusters the hierarchical clustering produced clusters of higher precision scores than the results of the PCA. This means, the topology of clusters computed by the hierarchical clustering algorithm possesses a higher similarity to the originating taxonomy of DBpedia than the clusters computed by the

PCA. In both cases, the taxonomic recall and the harmonic mean (F-measure) between recall and precision decrease monotonously when the amount of clusters decreases. In contrast to taxonomic recall ratios, the ratios of taxonomic precision oscillate. By ignoring extreme cluster results with high or low amount of clusters, the remaining local optima are 32 clusters in case of hierarchical clustering and 23 clusters in case of PCA. Figures 6.5 and 6.6 illustrates two rendered PCA and hierarchical clustering results, which were rated best according to the local optima of taxonomic precision values in Figure 6.4.

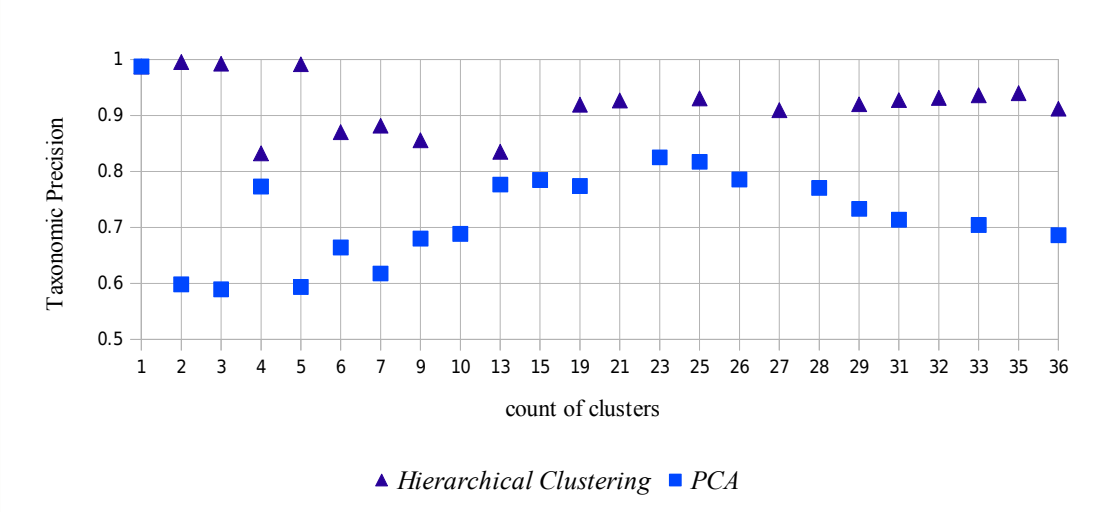


Figure 6.4: Taxonomic precision ratios of PCA and hierarchical clusterings.

It can be observed that the PCA computed clustering, which is rated best and is rendered in Figure 6.5, subsumed a heterogeneous (in terms of the originating taxonomy) set of classes (i.e., Scientist, TennisLeague, Language, etc.) as SiteOfSpecialScientificInterest[5]. In contrast, the clusters computed by the hierarchical clustering (see Figure 6.6) contain homogeneous (in terms of the originating taxonomy) sets of classes. Each class representing a cluster is also a super-class of a cluster's contained classes in the originating taxonomy. The best rated hierarchical clustering separated the taxonomy into a higher count of clusters than the PCA did. In addition, the average cluster size differs less than compared to the PCA-computed clustering.

In terms of complexity, the computation and labeling of clusters based on principle

[5]In order to increase readability, the prefix `dp-ont` was dismissed within the rendering of Figures 6.5 and 6.6.

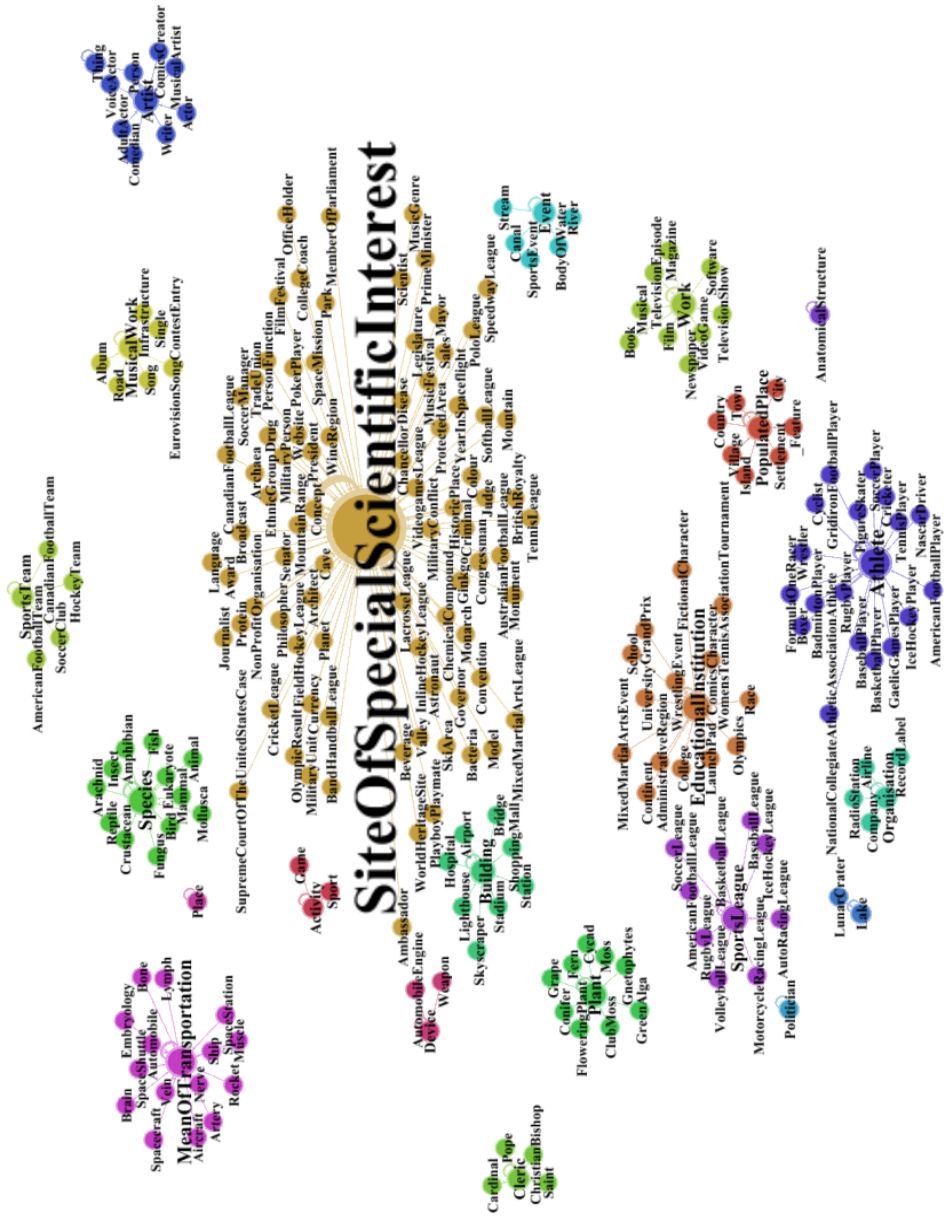


Figure 6.5: PCA generated clustering of DBpedia classes.

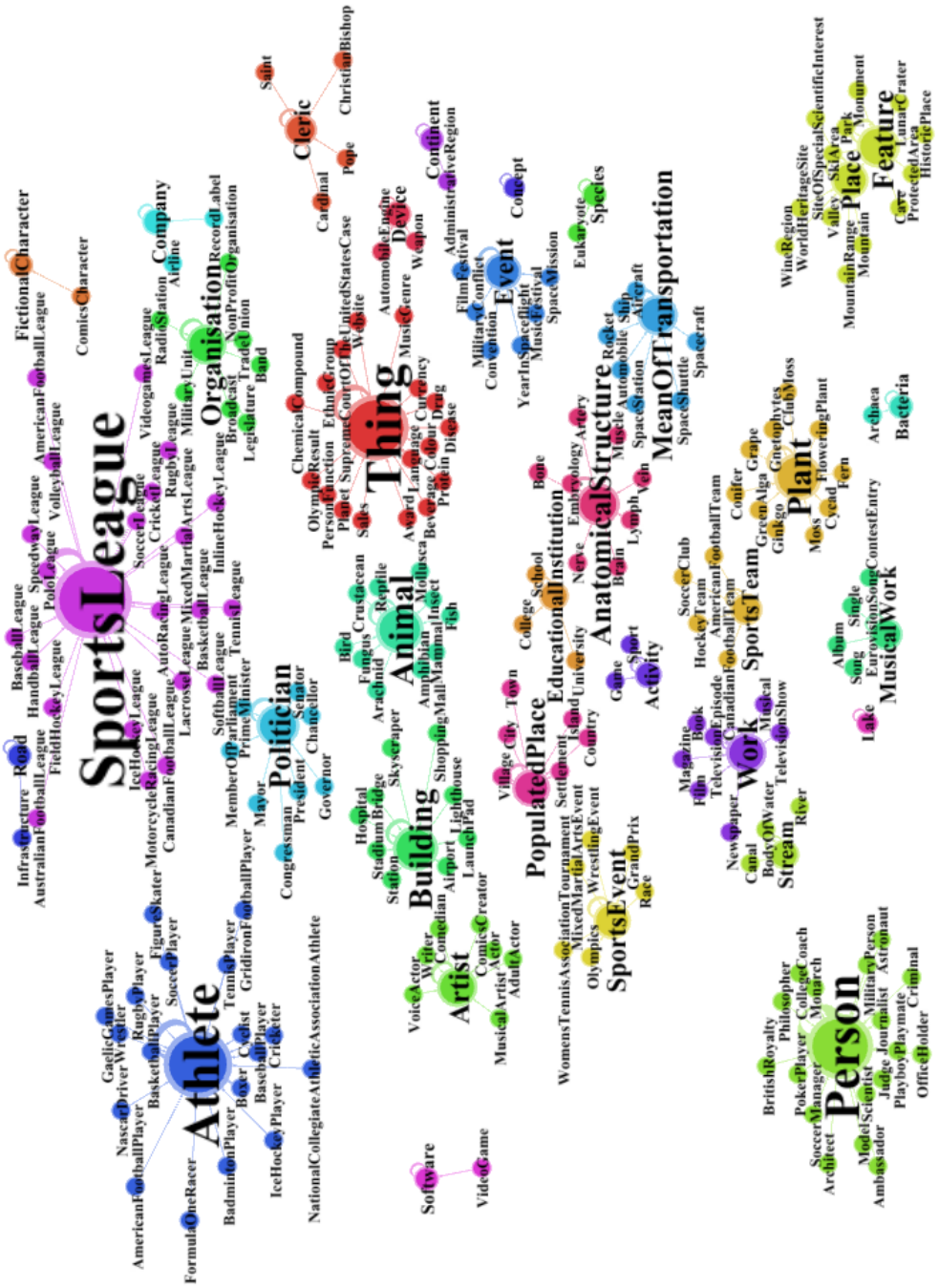


Figure 6.6: Hierarchical clustering based generated clustering of DBpedia classes.

components is easy to implement. The comparison of clusters, which were calculated by hierarchical clustering and PCA, reveals that the hierarchical clustering produces clusters of higher quality than the PCA.

The presented approaches assume a materialized transitive closure of `rdf:type` statements within the RDF graph. In terms of the conditional probability matrix M_{COPR}, this assumption assures values of either zero or one within the upper left triangle representing ancestors. If this assumption fails for an RDF graph, the selection of representatives depends on the existing distribution of `rdf:type` relations between instances and classes within each cluster. In extreme cases, in which no transitivity of `rdf:type` statements is materialized, the described approach is unable to cluster any classes of the RDF graph. In such a case, the correlation matrix is a diagonal matrix, in which values of one only occur on the diagonal where $i = j$.

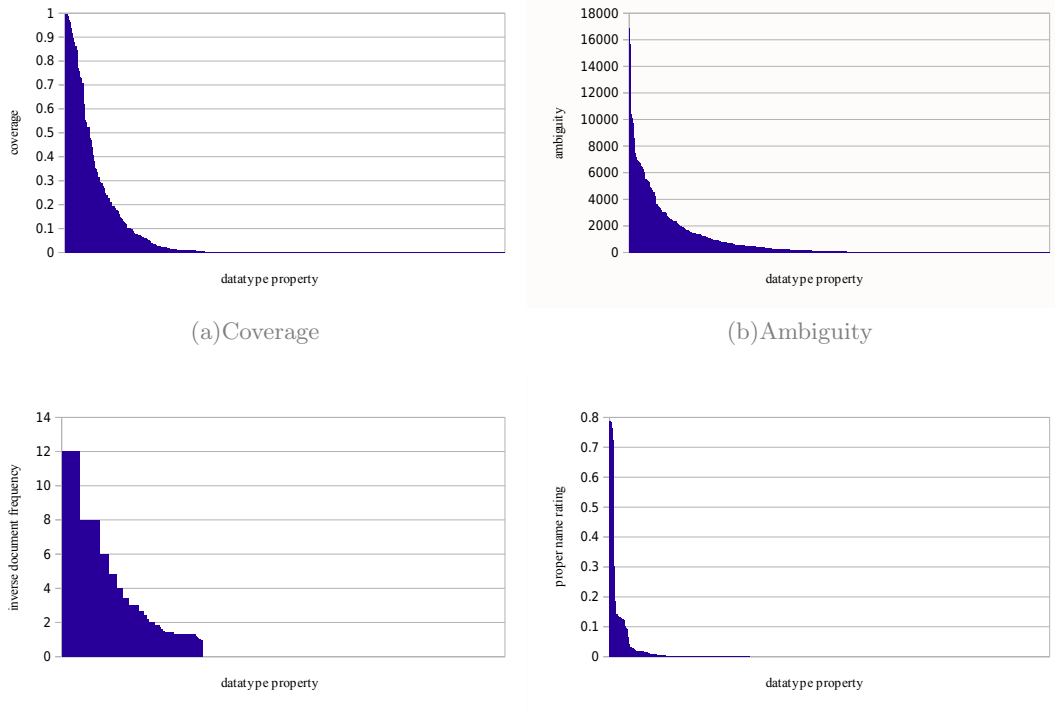
The idea underlying the Semantic Web encourages mixing hierarchies from various vocabularies. Within such mashups and the transitive closure of classifications, the described clustering approach returns a meaningful summarization of classes. In terms of the DBpedia, for example, the classes `dbp-ont:Person` and `foaf:Person` are clustered equally by the hierarchical clustering.

6.9.3 Proper name mining in DBpedia

Instances within the DBpedia are described by using around 500 different datatype properties. An analysis was conducted to describe the underlying distribution of each datatype property. The goal is to decide which datatype property is suitable to be used within an entity recognition task. Based on data from Reuters News corpus, Figure 6.7 illustrates histograms about the distributions of ambiguity, coverage, document frequency, and proper name ratings. The ordering of each histogram is descending in terms of frequency values. The analysis of these histograms in Figure 6.7 reveals that each rating follows a power-law distribution. Hence, the upper left part of each histogram indicates datatype properties of proper name values. Only to a few datatype properties possess combined proper name ratings near 1.0.

The following matrix in Table 6.1 lists correlation coefficients that describe the linear dependency between *ambiguity*, *coverage*, and *idf*. It can be observed that no linear correlations between these ratios exist on the DBpedia dataset. Hence, the factor combination of **ambiguity**, **coverage**, and **idf** does not deform average distributions and therefore preserves the underlying frequencies.

By using proper noun ratings, Table 6.2 lists top k datatype properties for places, works, persons, and companies. The two most clear and commonly used datatype properties are `rdfs:label` and `foaf:name`.



(a)Coverage (b)Ambiguity

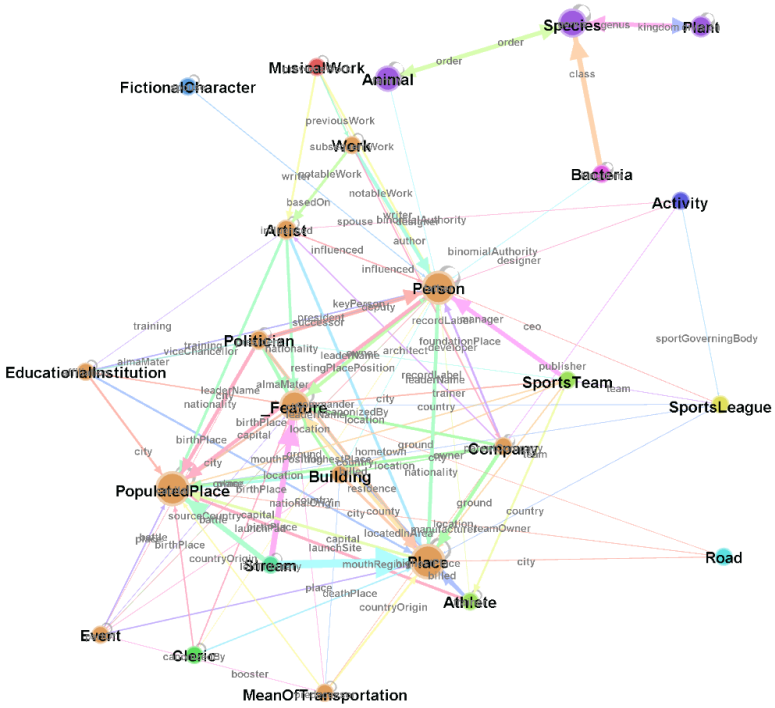
(c)Inverse Document Frequency (d)Proper name rating

Figure 6.7: Histograms of ambiguity, coverage, IDF, and proper name ratings.

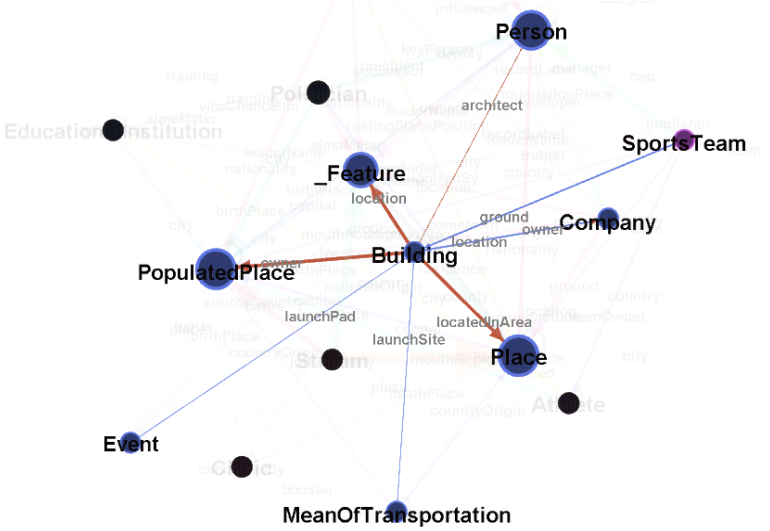
	ambiguity	**coverage**	**idf**
ambiguity	1.0	0.02	0.05
coverage	0.02	1.0	-0.01
idf	0.05	-0.01	1.0

Table 6.1: Correlation matrix about ambiguity, coverage, and idf.

The resulting insights of this study are that the datatype properties `rdfs:label` and `foaf:name` are good candidates for being used by a NER. In addition, it was a bit surprising that the person naming properties of the FOAF vocabulary were rated near zero (`foaf:nick`, `foaf:familyname`, etc.).



(a)Markov chain between DBpedia classes.



(b)Markov chain starting from `dbp-ont:Building`.

Figure 6.8: Markov chains on object properties.

type	datatype property	coverage	ambiguity	idf	rating
dbp-ont:Place	rdfs:label	1.00	1.22	0.96	0.79
	foaf:name	0.86	6.66	0.96	0.12
	geo:long	0.65	16.32	2.00	0.08
	geo:lat	0.65	21.93	2.00	0.06
	dbp-ont:postalCode	0.18	15.23	2.18	0.03
dbp-ont:Work	rdfs:label	0.77	1.22	0.96	0.61
	dbp-ont:review	0.14	1.42	3.00	0.30
	foaf:name	0.99	6.66	0.96	0.14
	dbp-ont:releaseDate	0.35	37.68	1.41	0.01
	dbp-ont:aSide	0.00	3.83	6.00	0.01
	dbp-ont:bSide	0.03	8.79	1.60	0.01
dbp-ont:Person	rdfs:label	0.99	1.22	0.96	0.78
	foaf:name	0.89	6.66	0.96	0.13
	dbp-ont:birthDate	0.73	13.53	1.41	0.08
	dbp-ont:birthName	0.03	4.36	12.00	0.08
	dbp-ont:deathDate	0.22	17.10	1.41	0.02
dbp-ont:Company	rdfs:label	1.00	1.22	0.96	0.79
	foaf:name	0.96	6.66	0.96	0.14
	dbp-ont:slogan	0.11	12.66	12.00	0.11
	dbp-ont:revenue	0.18	87.35	8.00	0.02
	dbp-ont:fate	0.05	63.08	12.00	0.01

Table 6.2: Proper name ratings for common classes in DBpedia.

6.9.4 Learning a Markov chain from DBpedia

In order to extract knowledge about proper relationships between classes in RDF graphs, an experiment was conducted that transformed object properties of a uniform sample of 100 instances of each class of the DBpedia into a Markov chain representation. Table 6.3 illustrates an extract of the resulted Markov chain. When assuming a subject of type `dbp-ont:Building`, the table lists the probabilities that this subject possesses an object property as transition to a target object of a certain class.

Figure 6.8 illustrates two graphical representations of the resulting Markov chain. In Figure 6.8a, a complete overview of relationships between classes is rendered as graph. Here, the width of edges represents the degree of probability. Figure 6.8b highlights outgoing and incoming relationships of class `dbp-ont:Building`. Experiments on the

transition	target	probability
dbp-ont:architect	dbp-ont:Thing	0.03
	dbp-ont:Person	0.03
dbp-ont:locatedInArea	dbp-ont:Thing	0.03
	dbp-ont:Place	0.02
	dbp-ont:PopulatedPlace	0.01
	geo:_Feature	0.003
dbp-ont:location	dbp-ont:Thing	0.22
	dbp-ont:Place	0.17
	dbp-ont:PopulatedPlace	0.11
	geo:_Feature	0.02
dbp-ont:operator	dbp-ont:Thing	0.03
dbp-ont:owner	dbp-ont:Thing	0.09
	dbp-ont:Place	0.04
	dbp-ont:PopulatedPlace	0.03
	dbp-ont:Company	0.02
	geo:_Feature	0.01
dbp-ont:owningOrganisation	dbp-ont:Thing	0.08
	dbp-ont:Place	0.04
	dbp-ont:PopulatedPlace	0.03
	dbp-ont:Company	0.02
	geo:_Feature	0.01
	sum	1.00

Table 6.3: Outgoing transitions by object properties of dbp-ont:Building.

DBpedia dataset show that computing Markov chains is easy.

6.9.5 Matching regular expressions with DBpedia literals

In order to review the effectiveness of learning to align regular expressions with datatype properties, three regular expressions were matched against literal values of the DBpedia. It has to be considered that the DBpedia may contain incorrect datatype property values. Hence, only those datatype properties whose values match with coverage above 90% were taken into account. The decision of setting the threshold to 90% was based on a rough estimate on experimental results.

Date expression (e.g., $1981-02-11$) In terms of the DBpedia, the evaluation of this regular expression resulted in 49 different datatype properties.

```
(19|20)\d{2}-(0[1-9]|1[012]|[1-9])-(0[1-9]|[1-9]|[12][0-9]|3[01])
```

The table below lists five example datatype properties that contain a date syntax.

datatype property	ratio
dbp-ont:closed	1.0
dbp-ont:season	1.0
dbp-ont:introductionDate	1.0
dbp-ont:rebuildingDate	0.98
dbp-ont:demolitionDate	0.96

The remaining mismatches show the existence of errors of incorrectly formatted date values in DBpedia.

Floating point number expression (e.g., -34.03453) The evaluation of the following regular expression resulted in 124 datatype properties.

```
[-]?[0-9]+\.[0-9]+
```

The table below lists five example datatype properties that contain a floating number syntax.

datatype property	ratio
dbp-ont:width	1.0
dbp-ont:areaWater	1.0
dbp-ont:floorArea	1.0
dbp-ont:maximumElevation	1.0
dbp-ont:salary	1.0

2d point expression (e.g., 49.484447 7.735939) Only values of a single datatype property matched with the following regular expression.

```
[-]?[0-9]+\.[0-9]+  [-]?[0-9]+\.[0-9]+
```

The table below contains a single datatype properties that contain a syntax representing a two dimensional point.

datatype property	ratio
http://www.georss.org/georss/point	1.0

Finally, it can be concluded that for a given regular expression especially in RDF graphs like DBpedia the match of possibly hundreds of datatype properties involves the selection of relevant properties by mechanisms like SPARQL queries as it is going to be described in Chapter 8.

6.9.6 Labeling the ConLL 2003 Corpus

The automatic labeling of corpus data was performed by using proper names of instance from DBpedia and by annotating these to recognized references of semantic entities in the Reuters News corpus. The quantitative quality rating of this automatically produced training data was performed by comparing results of two classifiers, the first being trained on the ConLL labeling of Reuters corpus, the second being trained on the automatically generated training data. Because results of these classifiers are also influenced by the selection of features, Section 7.7 first explains the effectiveness of different feature settings, before evaluation results are presented in Section 7.9.7.

6.10 Summary and conclusion

Motivated by the Hypothesis H.1, this chapter described pre-processing methods on RDF graphs.

6.10.1 Summary

The presented pre-processors allow the extraction and selection of various kinds of feature models from RDF graphs and text corpora. Subsequently, these models are provided to the information extractors, which will be described in Chapter 7.

- Section 6.2 illustrated a relational database schema specialized on representing RDF graphs specifically for IE-purpose.
- The clustering approaches presented in Section 6.3 successfully achieved to reduce the amount of classes within RDF graphs to a manageable and distinct set of clusters that can be used in succeeding analyzes or IE-methods .
- In Sections 6.4, 6.5, and 6.6 the calculation of statistical models was explained. The presented approaches built upon graph and text corpora statistics. Finally, the application of these models to information extractors fosters the recognition of semantic entities in natural text (see the Contribution C.1).
- In Section 6.7, a method was presented that automatically learns which datatype properties contain values that match with passed regular expressions. This approach extends the Semantic Entity Recognition process.

- Section 6.8 presented an approach for automatically creating corpus data for training an entity classification model.
- Finally, Section 6.9 listed a series of experiments, which were conducted for investigating the effective use of presented pre-processors.

6.10.2 Conclusion

The presented rehashing techniques of RDF graphs and contained information resulted in contributing the following solutions listed in Section 1.3:

Contribution 1 covers the programmatic incorporation of RDF knowledge into the IE system. This goal was reached by developing the relational database model. The consolidation of contained information was performed in term of clustering correlating classes and creating meaningful statistics on distributions of datatype and object properties, occurrences in text corpora and the nature of proper names.

Contribution 2 relates to the Semantic Entity Recognition process. The implementation of contained information extractors, which utilize information from RDF graphs, is based on the performed pre-processing techniques resulting in statistical models.

Contribution 5 corresponds with the automatically labeling of training data described in Section 6.8.

Contribution 6 covers domain independence, which was achieved by the presented preprocessing approaches.

7 Processing the Semantic Entity Recognition

> The good of a book lies in its being read. A book is made up of signs that speak of other signs, which in their turn speak of things. Without an eye to read them, a book contains signs that produce no concepts; therefore it is dump.
>
> *(Umberto Eco, 1983, The Name of the Rose)*

Eco describes the challenge of reading as recognizing signs in text and resolving these signs with already known or yet unknown concepts. In this work, the challenge of reading is transferred to information extractors utilizing concepts from RDF graphs. Hence, an information extractor is an algorithm, in terms of Eco a machine reader, which resolves signs in text by utilizing the formal conceptualization of an RDF graph.

This chapter is related to the research Hypothesis H.1. First, Section 7.1 introduces information extractors and summarizes the presented approaches that utilize RDF graphs. In Section 7.2, the recognition of proper names is illustrated. Section 7.3 specifies the recognition of RDF literals in text. The recognition of instances is explained in Section 7.4. The resolution of ambiguously recognized instances is illustrated in Section 7.5. Section 7.6 presents relevance ratings to recognized entities. The classification of noun phrases with classes from RDF graphs is described in Section 7.7. Section 7.8 shows approaches for predicting object properties between recognized instances. With respect to Hypothesis H.3, Section 7.9 illustrates experiments performed on multiple test corpora that prove the quality and adaptability of the presented approaches.

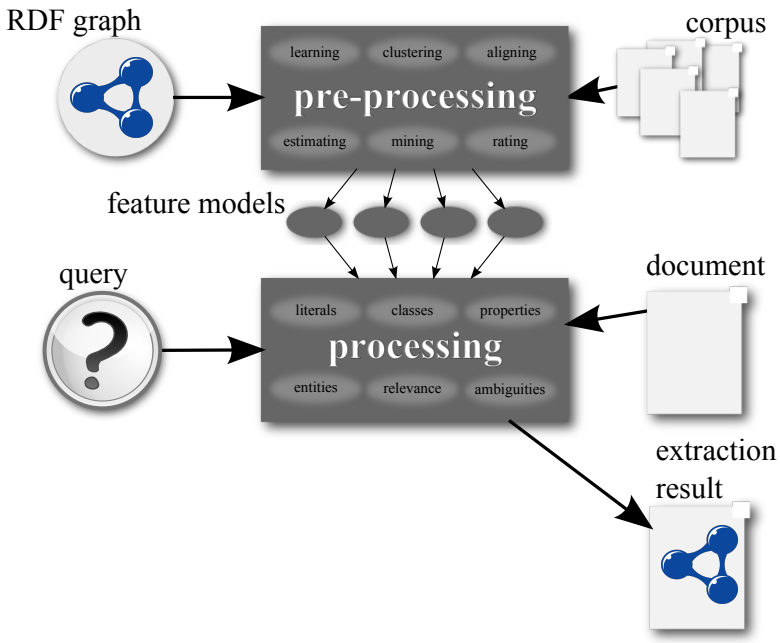


Figure 7.1: Information extractors within the Semantic Entity Recognition process.

7.1 Information Extractors

Information extractors are algorithms that implement specific IE tasks. Subject of this research is extending information extractors with following information sources:

- The originating natural language text document,
- Preliminary extraction results from preceding information extractors,
- Knowledge from pre-processed feature models (see previous Chapter 6),
- Specifying SPARQL queries that filter information (see Section 8.5).

In consequence of the RDF-based IE architecture, the output of the series of information extractors is serialized in RDF format. Figure 7.1 is part of the holistic Semantic Entity Recognition process, which is illustrated in Figure 5.7. The presented processing approaches depend on the pre-processing layer, which is illustrated in Figure 6.1. Finite-state transducers provide a framework for implementing information extractors (see Section 2.2.2). Consequently, the Semantic Entity Recognition process, which is outlined in Section 5.3, is designed as finite-state cascade consisting of a series of information extractors. On the basis of the Hypothesis H.1, each information extractor

utilizes RDF knowledge from the underlying RDF graph in form of the provided feature models. In the remainder of this chapter, the following information extractors will be presented (see also Section 5.3):

1. Filtering text for proper names (see Section 7.2)
2. Spotting text for datatype property values (see Section 7.3)
3. Linking named entities to formal instances (see Section 7.4)
4. Resolving ambiguous semantic entities (see Section 7.5)
5. Rating relevance of semantic entities in text (see Section 7.6)
6. Classifying semantic entities (see Section 7.7)
7. Predicting object properties between semantic entities (see Section 7.8)

In Section 7.9, these information extractors are evaluated, on behalf of five document corpora and three RDF graphs, for verifying the demanded adaptability of the Hypothesis H.3.

7.2 Filtering text for proper names

Natural language text documents may contain large amounts of words. RDF graphs may contain millions of names describing instances. Hence, the recognition of RDF literals in text as named entities requires the application of filtering mechanisms. These filters should remove irrelevant and noisy words and therefore reduce the overall amount of text segments in string comparisons. Section 5.1 describes NER depending on the recognition of proper names. Hence, the underlying role of nouns is used for creating an initial description of proper names. Figure 7.2 illustrates the general idea of filtering text segments by recognizing noun phrases.

The following text-based features were used to describe phrasal structures in text:

1. The current word,
2. The POS tag of that word (see Section 2.2.2),
3. The cases of its letters (upper, lower, capitalized), and finally
4. A window function that selects a context of previous and succeeding three words with corresponding POS tags.

The example below provides an impression on how the featured data looks like that is passed to the classifier.

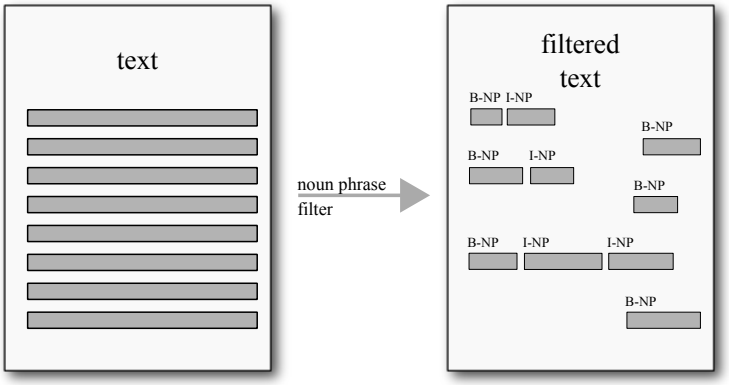


Figure 7.2: Filtering plain text for noun phrases.

Example 7.1 (*Noun phrase chunking*)
Assuming the sentence below:

"Barrack Obama was born in Honolulu."

Features *used to describe the occurrence of the word "Obama" in text are:*

***word content**: Obama*
***POS tag**: \NN*
***case of word**: upper*
***word window**: -1:word:Barrack; +1:word:was; +2:word:born; +3:word:in*
***POS window**: -1:pos:\NN; +1:pos:\vrb; +2:pos:\vrb; +3:pos:\in*

On the basis of text chunking techniques and a CRF (see Section 5.4.8), a proper name filter was trained to recognize noun phrases (see Section 2.2.2). It was decided to choose a CRF-based model, because in the ConLL 2000 text chunking challenge[1] the use of CRFs achieved best results. The implementation of the Mallet library is used [McCallum, 2002]. Resulting noun phrase chunkers are trained on the English ConLL 2000 and the German Tiger Brants et al. [2002] corpus.

The proper names that were recognized in the text are further processed by succeeding information extractors:

- When spotting a text for datatype property values describing proper names (see Section 7.3), filtering noun phrases reduces the amount of string comparisons.

[1] Please refer to text chunking results listed at `http://www.clips.ua.ac.be/conll2000/chunking/`

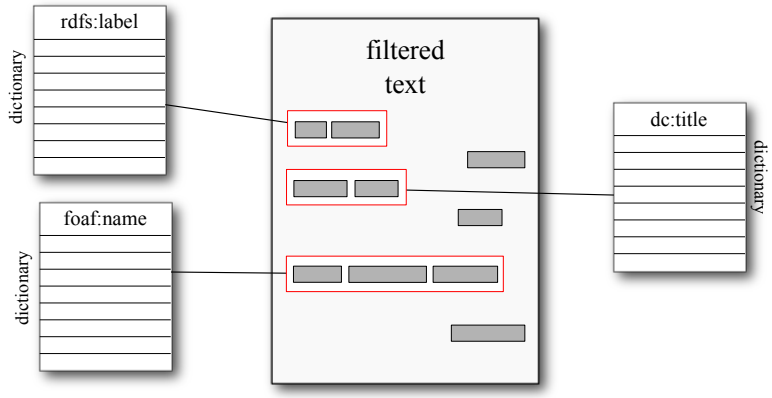


Figure 7.3: Like dictionaries in linguistics, datatype properties subsume literal values.

- The classification of semantic entities (see Section 7.7) is implemented as classifying noun phrase segments in text with RDFS classes or datatype properties. Here, the goal is to classify noun phrases that do not match with yet known literal values in the RDF graph.

The experiments described in Section 7.9.3 confirm the filtering effects of noun phrases in text. Some aspects of this work on filtering text for proper names were already published in [Adrian and Schwarz, 2011].

7.3 Spotting text for datatype property values

The recognition of datatype property values in text involves a comparison between a text (or a list of filtered noun phrases) and a list of literal values from the RDF graph. As described in Section 2.2.2 the current state-of-the-art in NER is determined by the use of dictionaries. Figure 7.3 illustrates the similarity between literal values of datatype properties and dictionaries. The substitution of dictionaries with datatype properties adapts the NER to the nature of names and concepts occurring in the underlying RDF graph (see hypotheses H.1 and H.3).

RDF graphs such as the DBpedia[2] may consist of hundreds of millions of literal values. This large number requires the application of scalable string comparison mechanisms. Approaches querying the database for matches on each noun phrase are impractical, because of the large amount of resulting string comparisons. On the opposite, hashing all literal values in memory is estimated as too memory consuming [Mendes et al., 2011].

[2]The query `select count(?label) where {?s ?p ?label. FILTER (isLiteral(?label))}` counts 105 019 913 literal values in DBpedia 3.6. Linked geo data counts 111 083 681 literal values.

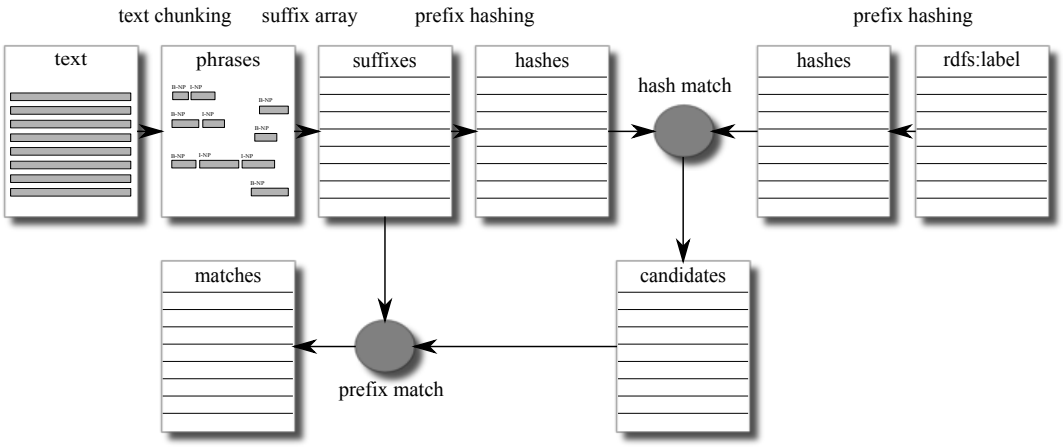


Figure 7.4: Algorithm for matching text with datatype property values.

Hence, Algorithm 7.1 is implemented for spotting RDF literals in text [Adrian and Schwarz, 2011]. Figure 7.4 illustrates the data flow.

Algorithm 7.1 (*Matching text with datatype property values*)

Input: *A text and an RDF graph, which is represented as described in Section 6.2, a parameter* $length_{prefix}$*, and datatype properties, whose values should be recognized.*

Output: *A list of named entities that match with literal values of the RDF graph.*

1. *Extract noun phrases from text by using a* text chunker*.*
2. *Represent these noun phrases in a* suffix array *(see Section 5.4.1).*
3. *For each entry of the suffix array, extract lexical prefixes of length* $length_{prefix}$*.*
4. *By using the hash function described in the Definition 6.2 in Section 6.2, compute numerical hash values for each prefix.*
5. *By executing the Query 7.1, request an ordered list of* candidate literals *with matching hashed prefix values from the list of datatype properties.*
6. *Compare the list of* candidate literals *with the* suffix array *via prefix matches. If the character-based comparison of an RDF literal with a prefix inside the suffix array is successful, the respective text segment is labeled as positive match.*

Query 7.1 (*Request datatype property values by hash matches.*)

```
-- For an input list of datatype properties and hash values, this query
-- returns an ordered list of matching literal values.
SELECT DISTINCT L.literal, L.index, S.predicate
FROM index_literals L, symbols S
WHERE ( S.predicate IN (
                -- <list of datatype properties>
) AND S.object = L.index AND L.prefix IN (
                -- <list of hashed prefix values>
 )) ORDER BY L.literal
```

Algorithm 7.1 spots RDF literals efficiently in a complexity of $O(|\text{suffix array}| + |\text{literals}|)$. It depends on stepping through both sorted lists. Based on the use of cursors[3] provided by modern relational databases, the local consumption of memory is reduced to a minimum. Example 7.2 illustrates the algorithm's workflow:

Example 7.2 (*Matching text with datatype property values.*)

"Quotations in the Frankfurt Stock Exchange reacted sensitively on the imprudent statement by Philipp Rösler."

- *Based on this sentence the noun phrase filtered suffix array, extracted prefixes and corresponding hash values are:*

Suffix	***Prefix of suffix***	***Hashed prefix***
Frankfurt ...	*Fra*	*70901*
Frankfurt Stock Exchange ...	*Fra*	*70901*
Philipp Rösler ...	*Phi*	*80209*
Quotation ...	*Quo*	*81579*
Stock Exchange ...	*Sto*	*83470*

- *Requesting the database (by Query 7.1) with these hash values, might result in a list of literals such as: "Frankfurt", "FraPort", "Philipp Hoschka", "Philipp Rösler", "Quotation", "Stock", "Stock Exchange".*
- *The prefix match between this result list and the suffix array results in these remaining entities: "Fankfurt", "Philipp Rösler", "Quotation", "Stock Exchange".*

[3] Cursors point to remote iterators on the database server side. See http://www.postgresql.org/docs/8.3/interactive/plpgsql-cursors.html

The selection of datatype properties, whose values to use for comparisons, can be triggered by using datatype properties with high proper name ratings (see Section 6.6). The matching strategy of the Algorithm 7.1 returns matching literal values from the RDF graph that are substrings of a given entry in the suffix array. A longest match strategy is applied in order to avoid the recognition of irrelevant or ambiguous instances whose datatype property values only match with substrings of the identified entity.

In the Example 7.3, the constraint to the longest match reduces the resulting list of matches to a single entry *One World Trade Center*. The cardinalities written in parentheses denote the count of distinct instances within the DBpedia that possess exactly these literal values. In the succeeding recognition of instances, restricting named entities to longest matches reduces the amount of ambiguously recognized instances (see Section 7.4).

Example 7.3 (*Longest match based comparison*)

The Algorithm 7.1 returns the following list of matching RDF literals from the DBpedia graph for the given noun phrase:

"One World Trade Center"

One World Trade Center	*(3)*
World Trade Center	*(34)*
World Trade	*(4)*
World	*(272)*
Trade Center	*(1)*
Trade	*(15)*
Trad	*(6)*
Center	*(931)*
Cent	*(13)*

In order to avoid collapsing homonyms in a text, the longest match strategy is implemented on the absolute character position of a named entity reference in the text. Hence, if within a text the smaller phrase "World Trade Center" occurs outside the longer phrase "One World Trade Center" the smaller phrase "World Trade Center" is not considered as being part of "One World Trade Center". The experiments described in Section 7.9.3 verify the quality of the presented approach.

7.4 Linking named entities to formal instances

After recognizing the literal values of a datatype property in text, the Semantic Entity Recognition involves the recognition of instances from the RDF graph. These instances possess the recognized datatype property and value. Figure 7.5 illustrates the recognition

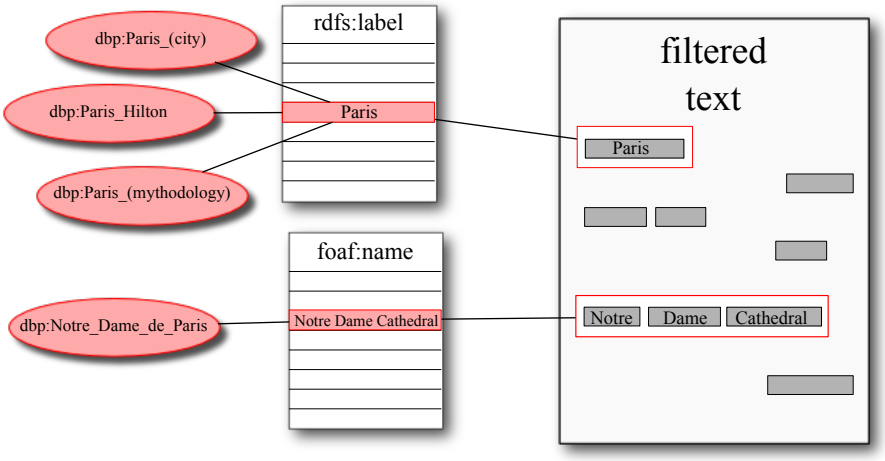


Figure 7.5: Linking named entities in text to instances in RDF graphs.

of a specific datatype property value in text. It results in creating a semantic link referring to the set of recognized instances.

Based on the representation of RDF graphs in the relational database schema (please refer to Section 6.2), for a given datatype property and value, the retrieval of instances can be implemented by querying the database with the Query 7.2. It returns a list of RDF triples consisting of subject, predicate, and object values. The predicate and object values correspond with the passed datatype properties and literal values. Returned subjects denote URI reference to instances that possess these property value pairs.

Query 7.2 (*Requesting subjects of recognized property-value pairs*)

```
-- For a given list of property value pairs, this query
-- returns a list of matching instances.

SELECT DISTINCT S.subject, S.predicate, S.object
  FROM symbols S
  WHERE (
    (S.predicate, S.object) IN (<list of property value pairs>)
  )
```

Example 7.4 illustrates the linking of named entities to formal instances.

Example 7.4 (*Linking named entities to formal instances*)
Assuming the following sentence:

> *"Notre Dame de Paris also known as Notre Dame Cathedral, is a Gothic, Catholic cathedral."*

The entity reference "Paris" can be resolved with several instances from DBpedia that possess "Paris" as literal value of the datatype property `rdfs:label`*, i.e.:*

- `dbp:Paris_(mythology)`*, a legendary figure of the Trojan War,*
- `dbp:Paris_(city)`*, the capital of France,*
- `dbp:Paris_Hilton`*, a female celebrity.*

This example shows that linking instances to named entities by matching property values may return multiple ambiguous instance references. Although it may be obvious for human readers, which of the listed instances was in the originating intention of this sentence, a machine reader needs additional formal background knowledge to perform any kind of instance resolutions. In Section 7.5, specialized algorithms for resolving these ambiguities will be presented.

Nevertheless, the presented approach for resolving instances is taken as a baseline in the evaluation in Section 7.9.4. The recognition of formal instances in text extends the originating definition of IE and more specifically NER. Semantic links associate named entities in the text with instances in the RDF graph. Hence, they extend the degree of information of named entities and therefore refine these as semantic entities. Linking additional information from RDF graphs to text segments confirms and supports the claim of Hypothesis H.1.

7.5 Resolving ambiguous semantic entities

In RDF, more than a single instance may be described by using the same datatype property value. For example, first and last names of persons are likely to be shared by multiple instances. In addition, polysemous words carry multiple meanings dependent on the context of use. For example, the word "crane" may be resolved to be either a name of a bird species or the name of a type of construction equipment.

In Computational Linguistics, this problem is covered by the Word-sense Disambiguation. Compared to other Word-sense Disambiguation approaches (see Section 2.2.2) utilizing external knowledge sources, this work investigates utilizing RDF graphs for resolving ambiguous entities.

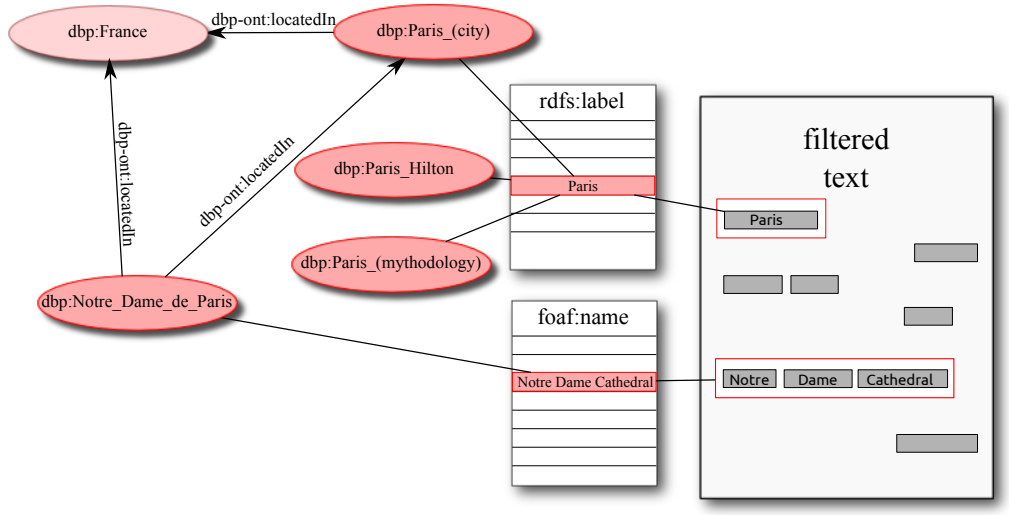


Figure 7.6: Relations between recognized instances indicate resolutions to ambiguities.

In consequence, when performing the Semantic Entity Recognition, ambiguities occur in cases where a single entity in the text can be resolved with different instances from the RDF graph. Figure 7.6 illustrates a scenario, in which the named entity "Paris" can be resolved with different instances of an RDF graph. The goal is deciding which instance from a set of ambiguously resolved instance candidates most likely refers to the intended real world entity. The set of ambiguously resolved instance candidates is defined as ambiguity group.

Definition 7.1 (*Ambiguity group*)

If a named entity in a text may be resolved with multiple instances from the RDF graph, the association of the named entity with the set of instances is referred to as ambiguity group.

The following example is derived from Figure 7.6. It illustrates the ambiguity group consisting of a set of instances from DBpedia the entity "Paris" might refer to.

Example 7.5 (*Ambiguity group*)

$$\text{"Paris"} \xleftarrow{\texttt{rdfs:label}} \begin{cases} \texttt{dbp:Paris\_(mythology)} \\ \texttt{dbp:Paris\_(city)} \\ \texttt{dbp:Paris\_Hilton} \end{cases}$$

The presented approach exploits the existence of two different contexts to achieve a disambiguation. On the one hand, the *textual context* is determined by an entity reference and its surrounding co-occurring entity references in text. On the other hand, the *graph context* is determined by the relationships between instances.

If the content of a text is covered by the domain of concern an RDF graph describes, it is likely that two co-occurring entity references in text are associated with a formal relationship between corresponding instances in the RDF graph. Example 7.6 illustrates this hypothesis more concisely.

Example 7.6 (*Resolving co-occurrences with formal relationships.*)

Two named entities "Paris" and "Notre Dame Cathedral" occur in this sentence:

> *"Notre Dame de Paris also known as Notre Dame Cathedral, is a Gothic, Catholic cathedral."*

Here, the semantic link of "Paris" might refer to either the tragic ancient Greek hero `dbp:Paris_(mythodology)`*, the capital of France* `dbp:Paris_(city)`*, or the female celebrity* `dbp:Paris_Hilton`*. The RDF graph, which is illustrated in Figure 7.6, comprises contextual background knowledge about the relationships between these instances:*

1. *A direct relationship associates* `dbp:Paris_(city)` *with* `dbp:Notre_Dame_de_Paris`*.*
2. *Both instances also share an indirect relationship by co-referring to* `dbp:France`*.*

For disambiguating instances, both kind of relationships indicate that the semantic link of "Paris" should refer to `dbp:Paris_(city)` *in this graph-based context.*

For a textual context, a respective graph context exists as being part of the RDF graph. Kleb and Abecker [2010] assume that, for co-occurring entities in the text, it is likely that RDF triples exist between them within the RDF graph. Hence, for each instance within an ambiguity group, a resolver may select the instance possessing the highest degree of connections to other co-occurring instances. The algorithm of Kleb and Abecker [2010] applies a spreading activation on an underlying RDF graph to find a minimal spanning tree (referred to as *Steiner graph*) that connects recognized instances. The shortest path between all kind of recognized instances may be very long. Especially paths between mistakenly recognized instances may populate the resulting *Steiner graph* with large numbers of irrelevant instances. Hence, Algorithm 7.2 was developed. Extending Kleb and Abecker [2010], the Algorithm 7.2 intends to find a minimal spanning tree between ambiguity groups instead of single instances. This reduces the size of the minimal spanning tree. Here, the minimal spanning tree between ambiguity groups is defined as neighborhood graph.

Algorithm 7.2 (*Linkage-based ambiguity resolution*)

Input: *An RDF graph G_{RDF} with a contained subset of instances whose datatype property values were recognized in a text.*

Output: *A list of resolved instances.*

1. *Create a* ***neighborhood graph*** $G_{neighbors}$ *with nodes* v_i^{ne} *that correspond with instances* $i_{ne} \in G_{RDF}$*, which are referred to by a* ***recognized named entity*** *ne.*
2. *For each ne compute an* ***ambiguity group*** $SET_{ne} \subset G_{neighbors}$*.*
3. *Sort all* SET_{ne} *by ascending size* $1.0 \leq n \leq max_{card(SET_{ne})}$ *.*
4. *By increasing* $1.0 \leq s \leq n$ *or until all* SET_{ne} *are connected in* $G_{neighbors}$*,*
   a) *For each node* $v_i \in \{SET_{ne} | card(SET_{ne}) = s\}$*, populate* $G_{neighbors}$ *with instances and object properties of the* ***symmetric concise bound description*** *(see Section 3.3) of* $v_i \in G_{RDF}$*.*
5. *For each* $v_i \in SET_{ne}$ *choose* v_i^{ne} *with the highest* ***connectivity*** *ratio.*

Based on a link analysis in RDF graphs, the following connectivity metrics are applied to disambiguate ambiguity groups (see Section 5.4.6):

- The *degree* of a node (see Definition 5.15)
- The *capacity* of a node (see Definition 5.16)
- The *hub* score of a node (see Definition 5.17)
- The *authority* score of a node (see Definition 5.17)
- The *PageRank* score of a node (see Definition 5.18)

Each of the presented node metrics assigns ratings to nodes. In terms of processing ambiguity group with the Algorithm 7.2, those node(s) reaching the maximum rating are taken for resolving the ambiguity. After disambiguating the set of recognized semantic entities, a rating referred to as decidedness denotes the remaining amount of ambiguity of recognized semantic entities.

Definition 7.2 (*Decidedness*)

Let recognized semantic entities consisting of an instance and a matching literal value be represented as tuple `(instance, literal)`*. Decidedness is defined as the ratio of the number of recognized URI references sharing the same literal value:*

$$decidedness(instance,\ literal) = \frac{1}{|(instance, literal)|}$$

In Section 7.9.6, the presented node metrics are evaluated on their use for resolving ambiguity groups. With respect to the Hypothesis H.1, the presented disambiguation approach verifies the potential of utilizing RDF graphs in IE. Based on linkage knowledge from RDF graphs about instances, mathematical ratings could be applied for disambiguating instances. Disambiguation is determined as hard problem in Computational Linguistics (see Section 2.2.2). Hence, utilizing the information from RDF graphs enhances IE approaches, which is claimed by the Hypothesis H.1.

7.6 Rating relevance of semantic entities in text

The Information Retrieval (IR) community refers to a document as relevant if it addresses a stated information need [Manning et al., 2008]. This can be transferred to the IE domain, in which a relevant instance relates to the current information need of a user. If an instance from an RDF graph remains outside this information need it is concerned as irrelevant. In general, two reasons can be revealed that lead to the recognition of irrelevant entities:

Irrelevant entity references For an entity reference in text, a correct instance is resolved from the RDF graph. However, the originating entity reference is not considered as relevant in terms of the textual context. For example, footers of Wikipedia pages mention the entity "Wikimedia Foundation, Inc." referring to the Wikimedia Foundation. Except for articles about Wikipedia and the Wikimedia Foundation itself, these entity references may be considered as irrelevant in terms of the overall topic of the actual article.

Instances resolved incorrectly Depending on the limited scope of content in used RDF graphs, it may occur that ambiguous literals are resolved incorrectly, because the RDF graph does not describe the correct instance. For example, the term "contact" occurs frequently in Web pages as label of a link that refers to the author's homepage, email address, or contact form. When using RDF data about movies from the DBpedia, the entity contact will likely be resolved as a feature of the same name.

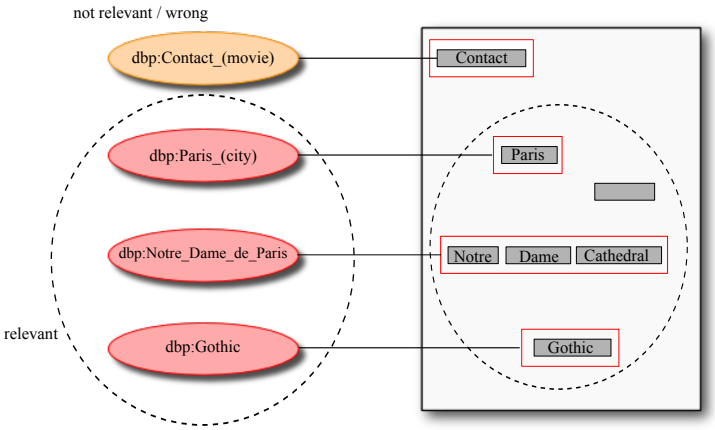


Figure 7.7: Some recognized semantic entities are more relevant than others.

In terms of RDF based IE, this information need is stated by the RDF graph representing the domain of concern. The presented approach uses structural knowledge from the RDF graph for rating the relevance of recognized instances in text documents. In the IR domain, a possible implementation of relevance applies statistical term distributions on the underlying document corpus. These metrics can also be applied for rating the relevance of recognized instances by considering the originating entity references. Combining the graph based and the document corpus based approaches seems promising for implementing an effective rating of relevance of recognized semantic entities.

Resulting from a concrete experiment, this example illustrates

Example 7.7 (*Recognized entities without relevance*)

"About the BBC, BBC Help, Contact Us"

Evaluating the Semantic Entity Recognition by using a documents corpus and an RDF graph from BBC music (see also Section 7.9.1) it could be revealed that in all documents the word "contact" was recognized as being the datatype property value of an equally named band. However, in the text, the word "contact" was part of the Web page header and meant to label a hyperlink to the Web site provider's email address. Hence, the resolved entity reference to a band was incorrect. The entity "contact" was completely irrelevant. More details on this study are published in [Adrian et al., 2010].

In IR relevance metrics assign relevance values to documents in the result list. This enables search engines to post-process result lists with rankings and filtering operations. Transferring this approach to Semantic Entity Recognition results in rating the relevancies of recognized entities for ranking purpose.

In Sections 6.4 and 6.5, a couple of statistical coefficients are proposed to rate terms and instances being parts of semantic entities. In addition, the *position in the text* of a token is also taken into consideration as coefficient for rating relevancies.

Definition 7.3 (*Position*)

The position of a semantic entity in the text is determined by the index of first occurrence in text.

$$pos_{t,d} = min(i|d[i] = t)$$

In summary, the following ratings are considered as suitable for ranking relevance of recognized semantic entities:

Text corpus-based ratings are computed by calculating statistics on word occurrences.

- **The entity's position in text:** The position of the first occurrence in the text.
- **The entity's term frequency:** The term frequency of the entity reference in the text (see Section 6.8).
- **The entity's inverse document frequency:** The overall inverse document frequency of the entity reference in the text corpus (see Section 6.9).

Graph-based metrics are computed by calculating statistics on node connectivities in RDF graphs. In consequence, the same set of algorithms (HITS, PageRank, node degree, node capacity) that was applied to disambiguate entities can be reused here.

- **The degree of a node:** The node degree of incoming or outgoing edges of the entity referent in the RDF graph (see Definition 5.15).
- **The capacity of a node:** The node's capacity of incoming and outgoing edges of the entity referent in the RDF graph (see Definition 5.16).
- **The node's authority rating:** Authority values of the entity referent in the RDF graph based on the HITS algorithm (see Definition 5.17).
- **The node's hub rating:** Hub values of the entity referent in the RDF graph based on the HITS algorithm (see Definition 5.17) .
- **The PageRank of a node:** Pagerank values of the entity referent in the RDF graph based on the identically named algorithm(see Definition 5.18) .

In Section 7.9.9, each of the presented rating is evaluated on its use for ranking recognized entities. Similar to Section 7.5, information from the RDF graph extends corpus based rating statistics for ranking recognized instances. In terms of the RDF graph, the notion of relevance is defined by the connectivity of the respective instance to other recognized instances in text. Again, the application of information from the RDF graph supports IE approaches, which is stated in the Hypothesis H.1.

7.7 Classifying semantic entities

In Section 6.8 an approach is described, which labels a document corpus with classes of recognized named entities. By using such a corpus as training data, a classifier is trained for predicting the classifications of entities by using classes from the RDF graph. In general, this classification goal corresponds with the traditional NER and classification where a predefined simple classification scheme consisting of location, person, and organization is used (see Section 2.2.2). In contrast to the traditional use of dictionaries, the proposed approach uses datatype properties from the RDF graph. In detail, a maximum entropy classifier (see Section 5.4.7) is applied to automatically learn types of semantic entities that share similar feature distributions. The following example illustrates the idea of labeling and classifying entities in text.

Example 7.8 (*Labeling and classifying entities in text*)

	`dbp:Place`	`dbp:Person`		`dbp:Person`
"'I love this girl from	***Sparta** ',*	***Paris***	*said to*	***Hector**."*

The following machine learning features were used to describe the occurrence of classes in text. They are categorized in context describing features, which contain information of an entity surrounding textual context, and content describing features, which contain information about the entity itself.

Context describing features — textual references to entities are embedded within a textual context of the surrounding sentence or list. Context describing features expose commonalities of these contexts. Different entities of the same type or literals of the same property may share these commonalities. Verbs, for example, may be good discriminators for classifying subjects or objects in sentences, such as "In 1981, Ben was born in Munich.".

- **Word Windows** — For a given word in a sentence, take the n words before and after this word the sentence. (e.g., for classifying "Ben" with a window of length 4, split "In 1981", Ben, "was born in Munich.")
- **Word n-grams** — Create conjunctions of n neighboring words. (e.g., "was born in Munich." results in "was born in", "born in Munich" 3-grams)

Neighboring known entities — For a word in sentence add neighboring known semantic entities in this sentence. (e.g., for classifying "Ben", substitute "Ben was born in Munich" with "Ben was born in LOCATION")

Syntactic patterns — Substitute words with matching patterns in syntax (e.g., substitute "In 1981, Ben " with "In DATE, Ben")

Content describing features — Literal values of entities of similar types or properties often share similar structural patterns in their content. Simple examples are strongly structured properties such as addresses, dates, or numeric values. But in some cultures even person names share similar characteristics as shown by the following examples: **O'**Brian, **Mc**Donald, **Iben** Fasid, Obradov**ic**, **van** Elst ....

Prefixes/Suffixes — Add the first or last n characters of a word as a new feature.

Syntactic patterns — Substitute words with matching patterns in syntax (e.g., it costs 10.53 $ — it costs `PRIZE`),

Combining the labeling approach presented in Section 6.8 and the feature extraction and classification approach presented here, enables the training of a maximum entropy classifier, which predicts the classes of recognized named entities. The quality of predicted classifications is investigated in Section 7.9.7. With respect to the claimed Hypothesis H. 1, the ability to use RDF graphs to create training data for an entity classifier shows the potentials of RDF for IE. Regarding the demanded increase of adaptability in the Hypothesis H. 3, the presented labeling and classification approaches are a great step towards adapting existing IE systems to completely new domains.

7.8 Predicting object properties between semantic entities

A text spoken in natural language refers to multiple named entities. Between recognized instances of these co-occurring entities, the existence of semantic associations can be assumed. Simple natural language sentences like illustrated in Example 7.9 consist of:

- A *subject* denoting a named entity,
- A *predicate* in terms of a verb phrase, and finally
- An *object*, which is again denoting a named entity.

Here, the association between both entities is determined by the verb.

Example 7.9 (*Facts*)

The following sentence refers to two entities, namely the author of this document and his favorite meal.

"Benjamin likes eating Bavarian veal sausage"

The sentence associates both entities by using a verb phrase, which determines sausages to be a marked preference of Benjamin.

Sentences may contain syntax structures such as sequences, nestings, or negations. Verbs may be substantiated or paraphrased by using the whole treasury of language. This complicates the algorithmic extraction of facts. Buitelaar et al. [2005] stated that most facts in text are implicit and remain "under the surface". Many facts can only be reasoned if background knowledge about an assumed common sense exists.

Natural language processing algorithms that automatically recognize and extract formal facts have problems coping with these language complexities (see Section 2.2.2). Therefore, the problem of fact extraction is far from being solved, yet. Oren et al. [2007] proposed the use of intrinsic structural knowledge from RDF graphs for suggesting predicates in terms of RDF triples. The intention of this work was recommending RDF properties from used vocabularies in order to further describing instances. Al-Rajebah and Al-Khalifa [2010] summarized existing approaches for extracting semantic relationships from Wikipedia. They concluded that the presented relation extraction approaches depend on either structured text input (e.g., tabular like infoboxes consisting of predicate-object pairs) or apply formal grammars on sentences in the plain text of each article. In this work, a hybrid approach is investigated for extracting facts between recognized semantic entities. Here, the structures of an underlying RDF graph are used for interpreting co-occurring entities in plain text. Two approaches were implemented based on the following models:

Markov chains Computing Markov chains on object properties between classes of instances was already described in Section 6.4.3. The idea underlying the use of Markov chains is to use the most probable object properties between the classes of two respective recognized instances for predicting object properties between these. For example, the Markov chain which was computed on the RDF graph from BBC music determines the object property `mo:memberOf` to most likely exist between an instance of the class `mo:MusicArtist` and an instance of the class `mo:MusicGroup`.

Similarity-based collaborative filtering This approach is inspired by similarity-based recommender systems [Koren and Bell, 2011] . Recommenders provide customers with recommended products. The recommendation of products is based on the existence of similar customers and the products they bought in the past. Here, the recommender

approach is transferred to IE by recommending object properties between recognized instances. Recognized instances in text can be referred to as customers. Predicate-object pairs of RDF triples can be referred to as products. Hence, at first, this similarity based collaborative filtering creates an instance/predicate-object matrix (formerly referred to as customer-product matrix) from recognized instances and existing RDF triples.

Example 7.10 (*Instance/Predicate-object matrix*)

Assuming the instances referred to as Benjamin, Michael, Kaiserslautern, DFKI to be recognized in a text and the following RDF triples to exist in the RDF graph (Namespace prefixes are not used in this example in order to facilitate matching each property later on with headers matrix columns):

```
Ben             livesIn     Kaiserslautern ;
                a           Person ;
                works       DFKI ;
                topic       AI .
DFKI            located     Kaiserslautern ;
                a           Organisation ;
                topic       AI .
Kaiserslautern  a           Town .
Michael         a           Person .
```

The RDF triples result in the instance/predicate-object matrix in Figure 7.8:

$$\begin{pmatrix} & a\_Person & a\_Town & a\_Org & lives\_KL & works\_DFKI & located\_KL & topic\_AI \\ Ben & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ DFKI & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ KL & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ Michael & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 7.8: Instance/predicate-object matrix of triples in Example 7.10.

Based on this instance/predicate-object matrix, the similarity between instances is computed by using standard similarity measures such as cosine similarities (see Definition 5.3) or correlation coefficients values (see Definition 5.8 of Pearson's product momentum coefficient). Figure 7.9 lists example similarity matrices derived from the matrix in Figure 7.8.

The matrix in Figure 7.9a shows that calculating simple cosine similarities between instances results in positive similarities even if these instances possess a different type. These similarities lead to a propagation of properties with strongly typed signatures to invalid instances with invalid classes (see matrix in Figure 7.10a, where the organization DFKI is recommended with `rdf:type foaf:Person`). Hence, the calculation of

$$\begin{pmatrix} Ben & 1 & 0.3 & 0 & 0.5 \\ DFKI & 0.3 & 1 & 0 & 0 \\ KL & 0 & 0 & 1 & 0 \\ Michael & 0.5 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} Ben & 1 & 0 & 0 & 0.4 \\ DFKI & 0 & 1 & 0 & 0 \\ KL & 0 & 0 & 1 & 0 \\ Michael & 0.4 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} Ben & 1 & 0 & 0 & 0.5 \\ DFKI & 0 & 1 & 0 & 0 \\ KL & 0 & 0 & 1 & 0 \\ Michael & 0.5 & 0 & 0 & 1 \end{pmatrix}$$

(a)cosine (b)positive pearson (c)cosine for equal classification

Figure 7.9: Similarity values for matrix in Figure 7.8.

similarity between instances should result positive values if both instances are classified similarly. In Table 7.9c results of an extended similarity calculation are presented, which allows only positive similarities between equally classified instances. Table 7.9b illustrates that positive or zero values of a correlation matrix also can be used as similarities. Although the correlation matrix in this example does not contain similarities between instances of different classes, this behavior does not hold for other datasets if they contain object properties, which are used independently from the classes of instances, frequently.

The Algorithm 7.3 illustrates the use of the instance/predicate-object matrix and a derived similarity matrix for adding new predicate-object pairs to the originating instance/predicate-object matrix.

Algorithm 7.3 (*Recommend property-value pairs for instances*)

Input: *An instance/predicate-object* `matrix`*, and similarity matrix* `similarity`*.*

Output: *An entailed instance/predicate-object matrix.*

```
def recommend(matrix, similarity) :
  for row in range(matrix.rows()) :
    for col in range(matrix.columns()) :
      value = matrix[row][col]
      if  value != 0 :
        for row2 in range (matrix.rows()) :
          sim = similarity[row][row2]
          value2 = matrix[row2][col]
          evidence = sim * value2
          value += evidence
      matrix[row][col] =  value
  return matrix
```

As a result of Algorithm 7.3, Figure 7.10 lists entailed instance/predicate-object matrixes derived from using different similary matrixes.

	a_Person	*a_Town*	*a_Org*	*lives_KL*	*works_DFKI*	*located_KL*	*topic_AI*
Ben	1	0	0.3	1	1	0.3	1
DFKI	0.3	0	1	0.3	0.3	1	1
KL	0	1	0	0	0	0	0
Michael	1	0	0	0.5	0.5	0	0.5

(a)cosine

	a_Person	*a_Town*	*a_Org*	*lives_KL*	*works_DFKI*	*located_KL*	*topic_AI*
Ben	1	0	0	1	1	0	1
DFKI	0	0	1	0	0	1	1
KL	0	1	0	0	0	0	0
Michael	1	0	0	0.4	0.4	0	0.4

(b)cosine for equal classification

	a_Person	*a_Town*	*a_Org*	*lives_KL*	*works_DFKI*	*located_KL*	*topic_AI*
Ben	1	0	0	1	1	0	1
DFKI	0	0	1	0	0	1	1
KL	0	1	0	0	0	0	0
Michael	1	0	0	0.5	0.5	0	0.5

(c)positive pearson

Figure 7.10: Predicted values for matrix in Figure 7.8.

The matrix illustrated in Figure 7.10a was calculated by using standard cosine similarities. It can be observed that it assigns several invalid predicate-object values to instances (i.e., second row representing `DFKI`). In contrast, the matrix in Figure 7.10b is calculated by restricting positive similarities by asserting equal classifications. Here, the matrix does not contain any recommendations that would violate signatures of object properties. The use of positive or zero valued correlation coefficients is shown in Figure 7.10c. In this example, the use of correlation coefficients seems to be more robust against invalid predicate-object values (i.e., second row representing `DFKI`).

The quality of recommended predicate object pairs is experimentally investigated in Section 7.9.10. Unfortunately, the calculation of cosine and correlation based similarity matrices is expensive in terms of CPU load. The reason is, that it requires at minimum a complexity of $O(n^2)$, where n denotes the size of the matrix. Hence, in Section 7.9.10 besides Markov chain based approaches the evaluation of recommender-based predicate object pairs could only be performed on the BBC music corpus. The BBC music corpus

contains the smallest amount of object properties (see Section 7.9.1). Nevertheless, the use of knowledge from RDF graphs for predicting object properties between two recognized instances in text indicates the general advantage of utilizing RDF in IE as claimed in Hypothesis H. 1.

7.9 Experiments

Up to this point, guidelines and algorithms are proposed for implementing information extractors that utilize knowledge from RDF graphs. The remainder sections describe results from performed experiment. Experiments evaluate the effectiveness and the efficiency of the presented approaches. Hereby is each experimental setting designed to provide an answer to a respective question. The following outline summarizes these questions and provides links to describing sections:

1. Do noun phrase filtering, prefix hashing, and suffix arrays provide a scalable basis for recognizing semantic entities? (see Section 7.9.3)
2. What is the baseline of naive semantic entities recognition? (see Section 7.9.4)
3. How does the proper noun rating of datatype properties support the recognition of semantic entities? (Section 7.9.5)
4. Which of the graph-based disambiguation approaches produces best results? (see Section 7.9.6)
5. Does automatically labeling corpus data allow training classifiers to predict classes of named entities? (see Section 7.9.7)
6. Can a disambiguation be performed based on entity classifications? (see Section 7.9.8)
7. Which relevance rating approach produces the best ranking of a list of recognized semantic entities? (see Section 7.9.9)
8. Is it possible to predict new object properties by using existing knowledge about object properties between recognized instances ? (see Section 7.9.10)

The experiments are performed based on labeled corpora, each corpus consisting of a document collection and a gold standard describing the set of semantic entities each document refers to. Each set of semantic entities is considered as reference, which has to be extracted by the Semantic Entity Recognition. A comparison between recognition results and gold standard references is implemented by matching the URI values of respective instances. In summary, each experimental test run consists of:

1. An RDF graph describing the domain of concern;
2. A document collection relating this domain of concern; and finally
3. A list defining URIs, which have to be recognized within a respective document.

7.9.1 Test Corpora

The following labeled corpora were created and used in conducted experiments:

Wikinews stories manually labeled with references to DBpedia instances

Wikinews is a free-content news source Wiki.[4] This corpus contains 99 small sized English documents with an average count of 262 words. As part of this work, in 2011, six pupils labeled each document manually with references to instances of the DBpedia. On average, each document is labeled with 11.5 instances.

Chapters from Project Gutenberg books manually labeled with DBpedia instances

Project Gutenberg is a public effort to digitize and archive cultural works for encouraging the creation and distribution of eBooks.[5] The corpus consists of 13 larger sized English documents with an average count of 7 968 words. In 2011, six pupils labeled each document manually with references to instances of the DBpedia. On average, each document is labeled with 17 instances.

Articles from Wikipedia automatically labeled with DBpedia instances

Wikipedia is a free, collaborative, and multilingual encyclopedia Wiki.[6] The corpus consists of 209 randomly chosen English articles with an average count of 606 words. In 2010, these documents were crawled. For each document, contained hyperlinks to other Wiki articles were resolved with corresponding DBpedia instances. The basis of this resolution is the existence of `foaf:page` properties in the DBpedia RDF graph linking the instances to the originating Wikipedia article. Finally, the hyperlinks were labeled with resolved instances. On average, each document is labeled with 18 instances.

Web pages from BBC music with existing formal RDF descriptions

BBC music is a Web portal about music artists hosted by the British Broadcasting Corporation (BBC).[7] The corpus consists of 12 464 small sized documents with an

[4] http://wikinews.org
[5] http://project-gutenberg.org
[6] http://wikipedia.org
[7] http://www.bbc.co.uk/music

average word count of 444 words. For each document, BBC provides a formal RDF description of contained instances including respective labels. These instances within the RDF data are used as gold standard. Thereby, on average each document is labeled with 8 instances.

RDF data from BBC music The RDF graph aggregated from the documents' RDF data consists of a total of 86 656 instances with assigned literals. A total of 104 120 distinct literals are to these instances by using 6 datatype properties. Most commonly used datatype properties are `foaf:name` with a count of 36 384 triples and `rdfs.label` with a count of 33 897 triples. Except links to the musicbrainz review portal anf `rdf:type` statements, a total of 20 object properties are in use. The object properties `mo:member_of` and `mo:member` are frequently used to relate music artists with music groups. The class hierarchy consists of 6 classes, 4 of them classify entities of the Music Ontology[8]. The remainder classifies blank nodes as birth and death dates.

Web pages from BBC nature with existing formal RDF descriptions

BBC nature is a Web portal providing information about nature and wildlife hosted by the British Broadcasting Corporation (BBC).[9] The corpus consists of 1174 medium sized documents with an average word count of 847 words. For each document, BBC provides a formal RDF description of contained instances including respective labels. These instances within the RDF data are used as gold standard. Thereby, on average each document is labeled with 32.5 instances.

RDF data from BBC nature The RDF graph aggregated from the documents' RDF data consists of a total of 8 722 instances with assigned literals. 9 490 distinct literal values exist within the graph. Literal values are assigned to instances by using 17 different datatype properties. Most commonly used properties are `rdfs:label` with 4 173 counts and `dc:title` with 4 769 counts. Remaining datatype properties are defined within the Wildlife Ontology vocabulary[10]. Except `rdf:type`, `owl:sameAs`, `rdfs:seeAlso`, 26 object properties are used to associate instance with each other. Again, the Wildlife Ontology is used. The class hierarchy consists of 38 classes, which are mostly defined by the Wildlife Ontology.

7.9.2 Evaluation metrics

In general, evaluation runs within the experiments result in a list of recognized instances represented as list of URI values. By using the reference list of URI values from the

[8] http://purl.org/ontology/mo/
[9] http://www.bbc.co.uk/nature
[10] http://purl.org/ontology/wo/

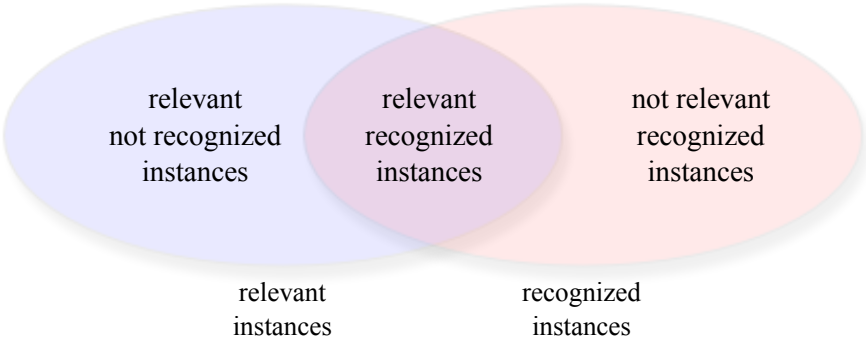


Figure 7.11: Sets of relevant and recognized instances.

gold standard of a respective corpus, extraction results can be rated according to well-known metrics such as recall and precision. These metrics are coined and used by the IR community Manning et al. [2008]. But also the IE community rates the quality of extraction results by using these metrics.

Figure 7.11 illustrates the correlation between recognized instances and instances, which are denoted as relevant by an underlying gold standard.

Based on the sets of relevant and recognized instances precision and recall can be defined as follows:

Definition 7.4 (*Precision*)

Precision describes the amount of relevant instances within the set of retrieved instances.

$$precision = \frac{|relevant\ recognized\ instances|}{|recognized\ instances|}$$

Definition 7.5 (*Recall*)

Recall describes the amount of relevant instances that could be recognized.

$$recall = \frac{|relevant\ recognized\ instances|}{|relevant\ instances|}$$

Definition 7.6 (*F1-measure*)

The harmonic F1-measure describes the harmonic mean between recall and precision.

$$F1 = \frac{2 * recall * precision}{recall + precision}$$

When analyzing extracted results in ranked lists, the IR community applies the following average precision metric.

Definition 7.7 (*Average precision*)

Average precision is the average precision about relevant instances inside the retrieved result list at rank r.

$$average\ precision(text) \quad = \quad \frac{\sum_{r=1}^{n} precision(i_r) * is\_relevant(i_r)}{\sum_{r=1}^{n} is\_relevant(i_r)}$$

The binary function is_relevant returns 1 if the current statement $stmt_r$ on rank r is relevant to the search.

$$is\_relevant(instance_r) = \begin{cases} 1 & if\ instance_r \in relevant\ instances, \\ 0 & else\ . \end{cases}$$

Definition 7.8 (*Mean average precision*)

Mean average precision averages the values of Average Precision over all text documents of the text corpus used in the evaluation setting.

$$mean\ average\ precision(corpus) \quad = \quad \frac{\sum_{j=1}^{|corpus|} average\ precision(text_j))}{|corpus|}$$

Within the following experiments, results are rated by using these metrics.

7.9.3 Accelerating the recognition of semantic entities

The goal of this experiment is to prove the effectivity and efficiency of the application of noun phrase filtering, prefix hashing, and suffix arrays to recognizing semantic entities. Utilizing large RDF graphs for recognizing semantic entities involves the matching of millions of literal values in natural language text. For this reason, the noun phrase filtering mechanism is proposed in Section 7.2. In order to show the potentials of filtering text for noun phrases, Table 7.1 presents frequencies of noun phrases and other words in

documents. The counts were calculated based on 100 documents, each chosen randomly from the English Wikipedia.

	words	**noun phrases**
text contains amounts of	75%	25%
	text	**noun phrases**
distinct words in	11%	89%

Table 7.1: Comparison of frequencies between noun phrases and other words in text.

It can be observed that only 25% of all the words of a text are determined as noun phrases, but exactly these noun phrases contain 89% of the distinct words of a text. Consequently, the filtering of noun phrases involves a reduction of 75% of recurring words occurring in these documents. Hence, the application of noun phrase filtering reduces the number of suffixes that have to be considered when building a suffix array. Still, the distinct remainder of these words form 89% of all distinct words in these documents. This ensures the conservation of a text's nature.

The suffix array algorithm, which is proposed in Section 7.3, reduces the length of prefixes by cutting them after a fixed count of characters. Based on all literals of the DBpedia RDF graph, the histogram of literals and their respective length in Figure 7.12 shows that the cut after 100 characters does not result in missing a significant number of literal values in subsequent string matching operations.

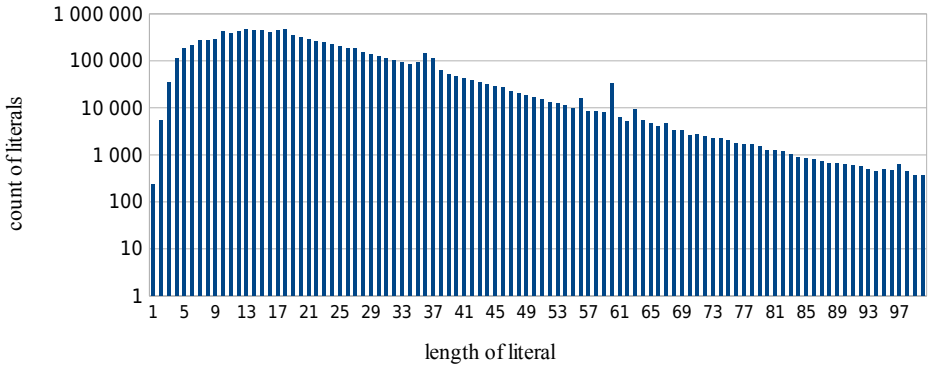


Figure 7.12: Histogram of length of literals in DBpedia.

In databases hash functions are commonly used for increasing the efficiency of comparisons in join operations. Hence, a hashing of literal values and suffix array entries is

built on the prefix of each literal or (filtered) word in a document. Figure 7.13 presents the number of results returned by the Query 7.2. The chart illustrates corresponding numbers of computed hash values on a scale of increasing prefix lengths.

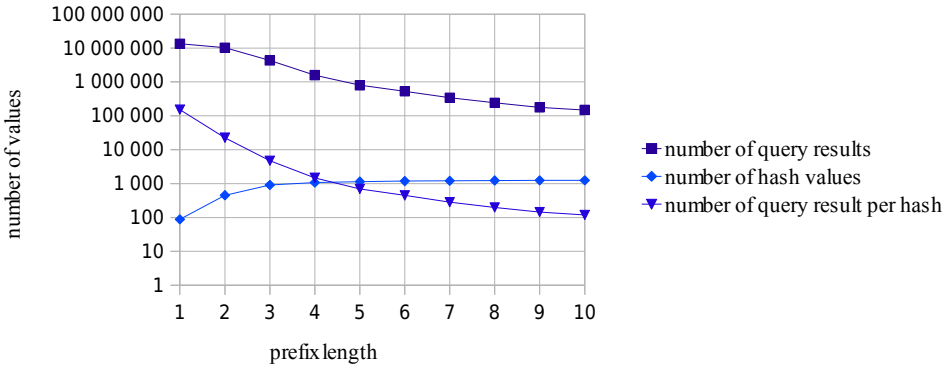


Figure 7.13: Count of query results in terms of prefix lengths.

It can be observed that the increase of prefix lengths involves an increase of hash values. Conversely, it reduces the number of query results. The average number of query results per hash value decreases but converges after a prefix length of four characters.

Table 7.2 lists response times of the Query 7.2 in terms of used string based or integer based hash values. Independent from the length of the prefix, the use of integers as hash values (which are computed by the hash function of the Definition 6.2) decreases response times constantly to a degree of 19% of the time needed by a string based hash function. This measurement was performed on a Laptop possessing a 1.8 Ghz Celeron with 3GB memory.

Based on the integer-based hashing of prefixes, Figure 7.14 illustrates the response times of the Query 7.2 and CPU times of a subsequent prefix comparison of query results and suffix array entries. Due to the reduced amount of query results, the increase of the prefix length involves a strong decrease of CPU time needed to compute the prefix comparison. Conversely, the query response increases. The accumulated time of the query response (computed remotely) and the comparison time (computed locally) decreases slightly when increasing the prefix length.

In summary, the performed experiments confirm the effectivity and efficiency of noun phrase filtering, prefix hashing with integer values, and suffix arrays to recognizing semantic entities, even in large RDF graphs such as provided by the DBpedia. In terms of prefix length, the best experiences are made by setting the length to four characters.

prefix length	string hash	integer hash	saving
1	28s	5s	19%
2	80s	16s	19%
3	193s	37s	19%
4	252s	49s	19%
5	276s	54s	19%
6	286s	56s	19%
7	288s	56s	19%
8	283s	55s	19%
9	273s	53s	19%
10	259s	50s	19%5

Table 7.2: Response times of the Query 7.2 of string or integer based hash values.

7.9.4 Naive recognition of semantic entities

In general, a baseline approach is determined by the simplest uninformed implementation that solves the problem to at least some degree. Transferred to Semantic Entity Recognition, the baseline for evaluating the quality of recognizing semantic entities is determined by the implementation described in Section 7.4. Because the longest match filtering does not utilize any kind of relational background knowledge from the RDF graph, apart from known literals, it is defined as baseline implementation. Table 7.3 shows that on the Wikinews corpus the longest match filtering strategy slightly increased precision ratios without influencing the recall.

matching strategy	precision	recall	F1
use all matches	0.05	0.73	0.09
filter longest match	0.08	0.73	0.14

Table 7.3: Recognized rates of longest match and all matches strategies.

By using this naive baseline implementation, Table 7.4 illustrates evaluation results for each of the applied test corpora.

It can be observed that baseline results are rated with values of high recall and low precision. Table 7.4 indicates that the amount of instances within RDF data and the size of documents have a negative influence on precision ratios.[11] The reason for recall

[11] This conclusion based on this data is of course only an estimate. It should be approved with more

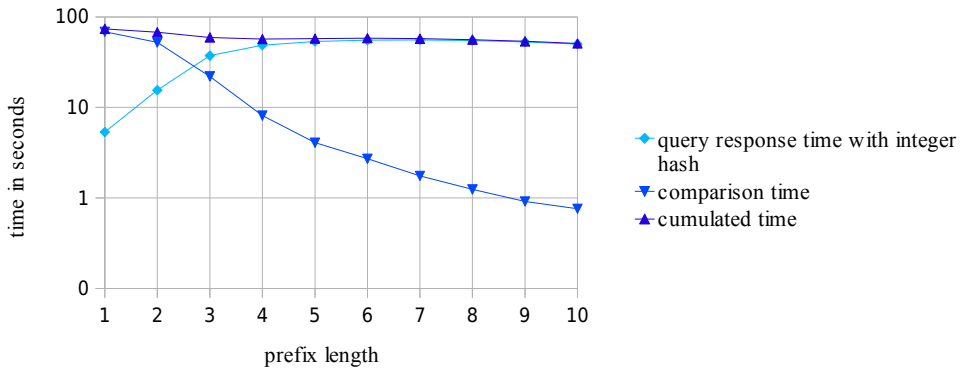


Figure 7.14: Processing times needed for querying and comparing literals.

corpus	$words_{avg}$	RDF graph	instances	precision	recall	F1
wikinews	262	**DBpedia**	8 013 297	0.01	0.74	0.02
wikipedia	606	**DBpedia**	8 013 297	0.004	0.65	0.01
gutenberg	7968	**DBpedia**	8 013 297	0.004	0.66	0.01
BBC music	444	**BBC music**	86 656	0.04	0.92	0.08
BBC nature	847	**BBC nature**	8 722	0.17	0.91	0.29

Table 7.4: Instance recognition results.

values below 1.0 is the variety of modified syntactic representations of named entities in natural language text. Generally, it is possible to implement comparison operators between literal values and text segments more efficiently by using approaches such as n-gram decomposition, edit distances, or stemmers. It was decided not to use such fuzzy comparators within conducted experiments, as they would distort resulting ratings when focusing on measuring the value of utilizing RDF data.

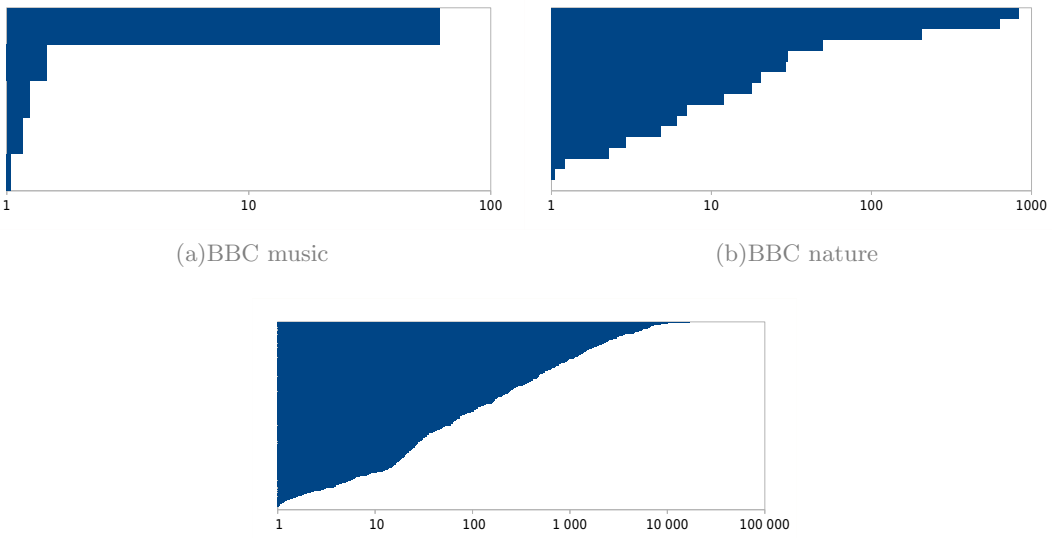
Despite the quality of results, the application of the baseline approach to recognizing semantic entities shows the simplicity and effectivity of utilizing RDF graphs (see Hypothesis H.1).

7.9.5 Filtering entity recognition by datatype properties

In RDF graphs, instances of classes are described by a variety of datatype properties. As described in Section 6.6 not all kind of datatype properties are suitable to be used as

details by experiments on more data. Unfortunately, labeled data rarely exists and labeling data is very expensive. Hence, such a formal evaluation goes beyond the scope of this work.

a basis for recognizing entities in text. Figure 7.15 illustrates histograms of ambiguity values of datatype properties in three RDF graphs. It can be seen that only the minority of all datatype properties possesses suitable ambiguity values near 1.0. In case of the DBpedia, some properties reach ambiguity ratios above 10 000.



(a)BBC music (b)BBC nature

(c)DBpedia

Figure 7.15: Histograms of ambiguity ratios of datatype properties.

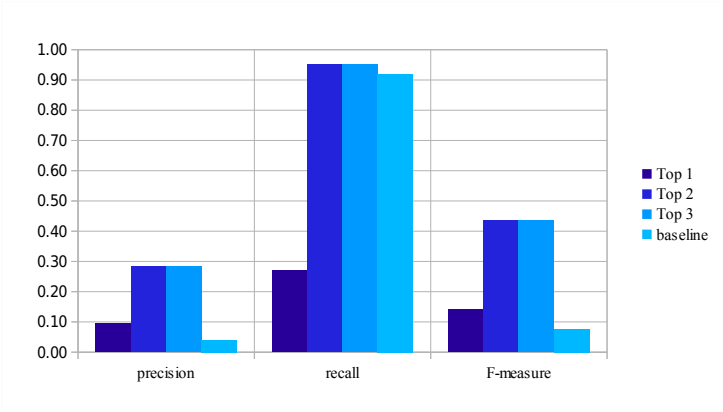
Using these datatype properties values in entity recognition algorithms results in a high amount of recognized instance candidates whose literals match with single text segments. This explains the low degree of precision of the presented baseline approach.

Hence, the ratings described in Section 6.6 (i.e., coverage, ambiguity document frequency) were applied to filter out appropriate datatype properties. Table 7.5 presents the top three ranked datatype properties for three RDF graphs. The ranking was calculated by aggregating the rating values of each datatype properties for all existing classes.[12]

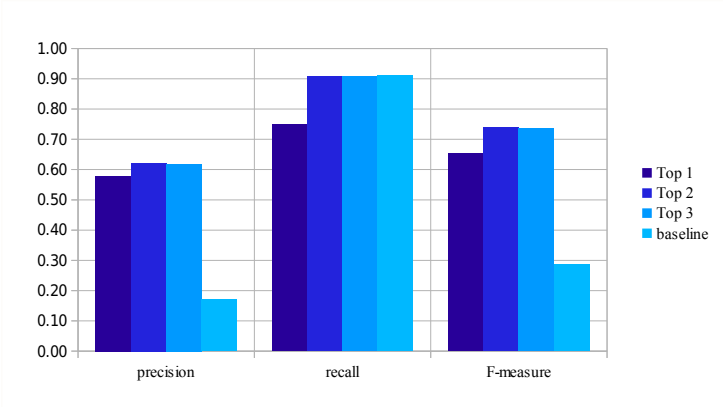
In terms of the DBpedia, it can be seen that the rating of `rdfs:label` is outstanding compared to the remaining properties. In both BBC datasets, the first two datatype properties possess similar ratings.

The three bar charts in Figure 7.16 illustrate the quality of extraction results when

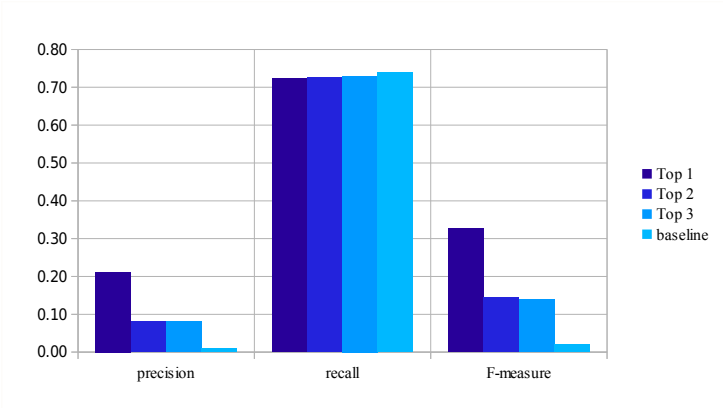
[12]The aggregation of rating values was calculated by this query: SELECT property, sum(rating) FROM proper_noun_rating GROUP BY property ORDER BY SUM(rating) DESC



(a)BBC music



(b)BBC nature



(c)DBpedia

Figure 7.16: Histograms of ambiguity ratios of datatype properties.

DBpedia	∑rating	BBC music	∑rating	BBC nature	∑rating
rdfs:label	23.16	ov:sortLabel	2.79	rdfs:label	6.35
foaf:name	3.5	foaf:name	2.39	dc:title	5.05
dbp-ont:review	1.1	dc:title	0.85	wo:speciesName	0.94

Table 7.5: Rankings of the top three datatype properties.

using either the top one, top two, or top tree datatype properties. These results were generated by using the corpora from BBC music, BBC nature, and Wikinews. Results by using the Wikipedia and Gutenberg corpus, which also depend on the DBpedia, look similar than results produced by the Wikinews corpus. In all datasets, the baseline, i.e., a recognition by using all datatype properties, always returns worst results in terms of F1-measure and precision. In terms of recall it can compete with the filtering approach. In Figure 7.16a, it can be seen the restricted use of `rdfs:label` does not cover the whole range of instances, which results in low precision and recall ratios. By adding `dc:title`, recall ratios increase and exceed the baseline. Figure 7.16b illustrates a similar scenario like before. In terms of the DBpedia in Figure 7.16c, it can be seen that an increase of datatype properties decreases precision ratios. The reason for this are the excellent rating values of the `rdfs:label` property as shown in Table 7.5.

This experiment approves the value of proper name ratings of used datatype properties within an RDF graph. By using the best-rated datatype properties, a higher precision of results can be recognized.

7.9.6 Graph-based disambiguation

In Section 7.5 a couple of graph-based algorithms were proposed to be used as a basis for resolving ambiguously resolved instances. The intention of the conducted experiments is to reveal, which of these algorithms performs best. This experimental settings are based on the Wikinews, the BBC music, and the BBC nature corpora. The following algorithms and parametric settings were evaluated:

- *degree:* The degree of a node.
- *auth:* The authority value of a node from the HITS algorithm.
- *hub:* The hub value of a node from the HITS algorithm.
- *hits+:* The sum of authority and hub values of a node.
- *hits(*):* The product of authority and hub values of a node.
- *cap:* The capacity of a node.

- *pr:* The PageRank values of a node from the equally named algorithm.
- *random:* A random selector, which is used as baseline approach.

The disambiguation was performed on results of the baseline of instance recognition, which used all kinds of datatype properties. This forced the occurrence of ambiguities in extraction results, because of the use of datatype properties with high ambiguity ratios.

Table 7.6 lists precision, recall, and F1 measure ratios, which were achieved by each algorithm on each corpus. The ratios express how well the algorithms selected the correct instance of an ambiguity group.

wikinews	**degree**	**hits(+)**	**auth**	**hits(*)**	**pr**	**cap**	**hub**	**random**
precision	**0.4**	0.39	0.39	0.38	0.38	0.26	0.23	0.20
recall	**0.51**	0.50	0.49	0.49	0.49	0.34	0.32	0.29
F1	**0.44**	0.44	0.43	0.43	0.43	0.30	0.27	0.24

BBC music	**degree**	**hits(+)**	**auth**	**hits(*)**	**pr**	**cap**	**hub**	**random**
precision	**0.085**	0.076	0.076	0.076	0.076	0.066	0.066	0.056
recall	**0.288**	0.248	0.248	0.248	0.248	0.178	0.178	0.138
F1	**0.131**	0.116	0.116	0.116	0.116	0.097	0.097	0.080

BBC nature	**hits(*)**	**degree**	**hits(+)**	**hub**	**cap**	**auth**	**pr**	**random**
precision	**0.827**	0.799	0.795	0.743	0.453	0.480	0.474	0.299
recall	**0.184**	0.184	0.181	0.179	0.106	0.094	0.087	0.079
F1	**0.301**	0.299	0.295	0.289	0.172	0.157	0.147	0.125

Table 7.6: Comparison of graph-based disambiguation algorithms.

On all three datasets, the applied graph-based algorithms performed better than the random baseline approach. On average, the resolutions performed by using the node degrees produce best results. In addition to this, the computation of a node's degree is very simple compared to the Eigenvalue-based computations of *PageRank* or *HITS* values. Finally, it can be concluded that based on the connectivity of recognized instances the best resolution of ambiguous instances can be achieved by using simply the node degrees as ratings. Consequently, the limits of the presented disambiguation algorithms are determined by the overall connectivity of an RDF graph. If relevant instances do not possess even a single relation to neighboring instances, the disambiguation will fail to resolve these from the ambiguity sets.

The conducted experiment shows that in RDF graphs the utilization of knowledge about links between instances allows the application of link analyzes based algorithms

for disambiguating recognized instances. This confirms the Hypothesis H.1.

7.9.7 Entity classification on automatically generated training data

The goal of this experiment is to examine to what degree the automatically labeling of raw corpus data with Semantic Entity Recognition results is suitable to train a named entity classifier. Hence, this experiment evaluates the quality of classification results of classifiers trained on automatically generated training data (see Section 7.7).

The corpus and dataset used for this experiment are:

The ConLL2003 corpus consisting of labeled news stories from Reuters. The labeled types of entities are: person, organization, location, and miscellaneous. The forth class miscellaneous is used as kind of residue class for all the other entity types in this corpus. More information on this corpus was already described in Section 6.9.1. The manually created labels of the ConLL2003 corpus were used for training a supervised classifier as reference model.

The DBpedia RDF graph as it is described in Section 6.9.1.

Aligning ConLL labels with DBpedia classes The goal is to use recognized semantic entities from the DBpedia dataset for automatically labeling the text of the ConLL2003 corpus. Results of this experiment reveal how a maximum entropy model trained on either the manually labeled or the automatically labeled corpus differs in its prediction accuracy. The first three classes of the ConLL2003 corpus were related to corresponding classes in the DBpedia ontology. Table 7.7 illustrates these corresponding classes.

class	frequency in RDF graph
`dbp-ont:Organisation`	147 889
`dbp-ont:Person`	363 751
`dbp-ont:Place`	462 349
`foaf:Person`	296 595
`foaf:Person` $\bigcap$ `dbp-ont:Person`	171 881
`foaf:Person` $\cup$ `dbp-ont:Person`	488 465

Table 7.7: DBpedia classes corresponding with ConLL2003 labels.

It can be observed that two individual but overlapping classes exist in the DBpedia representing persons. Based on the clustering approach presented in Section 6.3, these two correlating classes are merged. In the following analyzes the union of both is used.

class	nick	label	name	surname	gName	fName	title
dbp-ont:Organisation	0.076	0.995	0.951	0.001	0.006	0.000	0.004
dbp-ont:Person	0.022	0.992	0.922	0.449	0.467	0.002	0.027
dbp-ont:Place	0.01	0.996	0.888	0.000	0.000	0.000	0.000
foaf:Person	0.012	1.000	1.000	0.953	0.958	0.002	0.008

Table 7.8: Coverage of datatype properties used classes of the DBpedia.

Table 7.8 illustrates coverage ratios for datatype properties of each class.

The datatype properties `rdfs:label` and `foaf:name` cover most of the instances of all three classes. The distributions of the other datatype properties are dominated by instances of both types of persons. Based on the good ratings of the datatype properties `rdfs:label`, it was used as basis for recognizing matching named entities in the news stories of the ConLL2003 corpus.

Investigating the Independence of class labels In order to investigate the uniqueness of each class, Table 7.9 illustrates correlation coefficients between DBpedia labels and ConLL2003 labels (see Section 5.4.4).

		DBpedia			ConLL2003		
		person	**place**	**organ.**	**person**	**location**	**organ.**
DBpedia	**person**	**1.00**	-0.44	-0.59	**0.85**	-0.50	-0.44
	place	-0.44	**1.00**	-0.34	-0.28	**0.46**	**0.02**
	organ.	-0.59	-0.34	**1.00**	-0.49	-0.22	**0.64**
ConLL2003	**person**	**0.85**	-0.28	-0.49	**1.00**	-0.42	-0.37
	location	-0.50	**0.46**	-0.22	-0.42	**1.00**	**-0.02**
	organ.	-0.44	**0.02**	**0.64**	-0.37	**-0.02**	**1.00**

Table 7.9: Pearson correlation coefficient matrix.

Interestingly, it can be observed that the ConLL2003 labels representing organizations and locations share a higher degree of positive correlation than the other labels do. The same statistical pattern is valid for the automatically annotated labels from the DBpedia dataset. This statistical rationale indicates the ambiguous usage of location names. The reason is, that often news stories name sport teams similar to the countries and cities the team members play for (e.g., "Germany bet England 2:1"). In consequence, lower rates of precision and recall are expected when predicting locations.

Text corpora The maximum entropy classifier was trained on four versions of labeled training data:

Manual The original, manually labeled training set of the ConLL2003 corpus consists of 19 277 training instances.

Automatic This denotes the same training data as above, but it is now labeled with known semantic entities of the DBpedia database. This leads to a reduced amount of 8 495 training instances, because the DBpedia does not know all the entities labeled by hand.

Automatic+ By adding raw unlabeled data of the ConLL2003 corpus to the above training set extended the training set to 14 677 instances.

Automatic++ This corpus just consists of the automatically labeled raw data that resulted in 34 219 training instances without the original training corpus of the "Automatic" configuration.

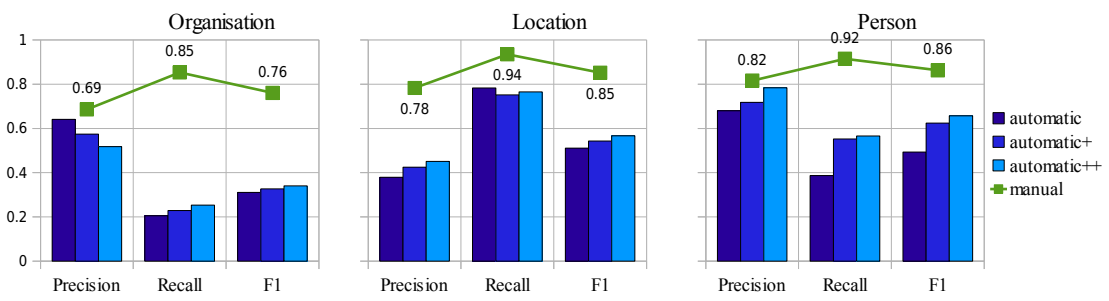


Figure 7.17: Accuracy of classification results on ConLL2003 test data.

Quality of classification results The quality of the classifier's results is shown in Figure 7.17. Precision, recall and F-measure values result from the comparison between the classifier's results and gold standard labels of the official ConLL2003 test corpus. Above each group consisting of three bars, the value of the classifier, which was trained on the manually labeled corpus, is printed as reference "upper line". Except the decreasing precision values of organizations, which correlate with decreasing or stagnating recall values of locations, it can be observed that the larger the training corpus is the better the expected results are. In case of persons and organizations the automatic labeling

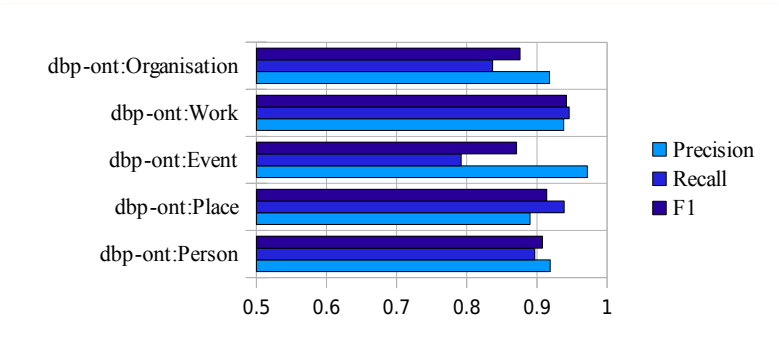


Figure 7.18: Accuracy of classification results in a 10-fold cross validation.

approach produces similar precision ratios like the manual labeled corpus. Unfortunately, the correlation between locations and organizations had a negative impact on the accuracy of recognition rates for locations and organizations.

In order to prove the consistency of the automatically generated labels, a 10-fold cross validation was performed on the largest training corpus (i.e., automatic++). Here, labels were added covering the classes `dbp-ont:Event` and `dbp-ont:Work` as they are described by the DBpedia ontology and are mentioned within the News stories. Figure 7.18 shows that all F1-values are exceed 0.8 and vary around 0.9 independently from the underlying classes. This is a strong evidence that the labeling generation produces consistent results and can be used to train classifiers.

Window and n-gram features

This investigation begins with a feature analyzes on the impact of window and n-gram features. The chart in Figure 7.19 presents harmonic F-measure values of a classifier that was trained on four different configurations of the ConLL 2003 corpus. Each axis represents a parameter setting consisting of a window size W and an n-gram conjunction n. The window sizes from 3 to 4 words before and after the entity candidate were evaluated. Within a window, the contained words are combined to partial n-gram sequences from length 1 to 4. By inspecting the values of the manually labeled corpus, it can be seen that results of a bag of words approach (that is n-grams of length 1) decrease in F-measure with an increasing window size. For n-grams of length 2 and 3, increasing the window size beyond 4 has no impact on the F-measure rates. In order to compute a limited number of features, we decided to use a default parameter setting of $W = 4$ and $n = 3$ without n-grams of length four.

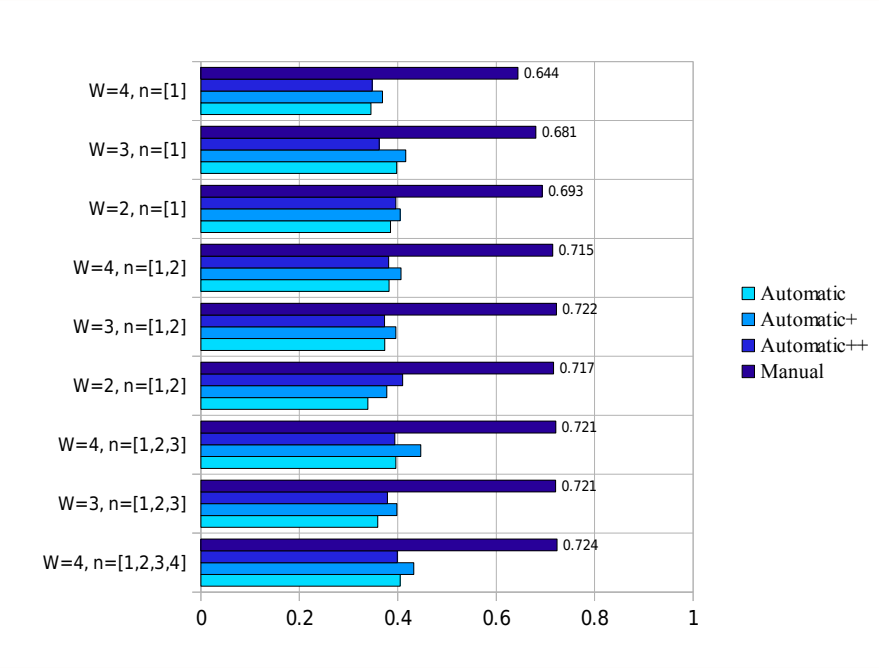


Figure 7.19: Comparison of window lengths (W) and n-gram conjunctions.

Context and content features

This investigation analyzes which feature type, context, content, or a combination of both performs best. Figure 7.20 presents the results of this evaluation. The reference values of the manual corpus shows that content features perform better than context features. A conjunction of both feature types results in a slight increase of quality.

Learning curves Analyzing the learning curves of classifiers being trained on either context or content features, it can be observed that the use of both feature types on manually labeled data results in the largest area under the curve. Both feature type curves indicate that content features are probably easier to learn than context features. The reason for this is that different news articles refer to similar topics. Within the 19 277 training examples of the original ConLL corpus, only 3 949 distinct persons, 1 464 distinct locations, and 2 665 distinct organizations are mentioned. Compared to this, the automatically labeled version consists only of 1 437 persons, 903 locations, and 531 organizations. This low degree of distinct values explains the dominant impact of content features on the classification accuracy.

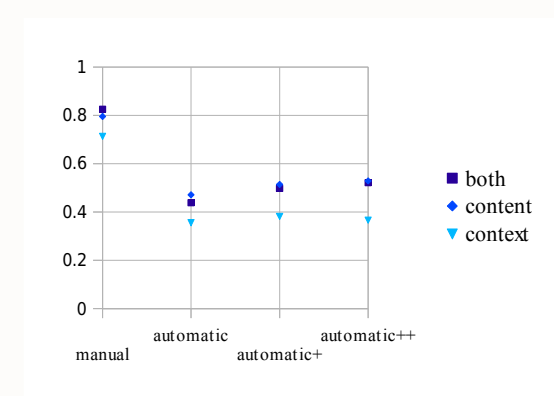


Figure 7.20: Comparison between content and context features.

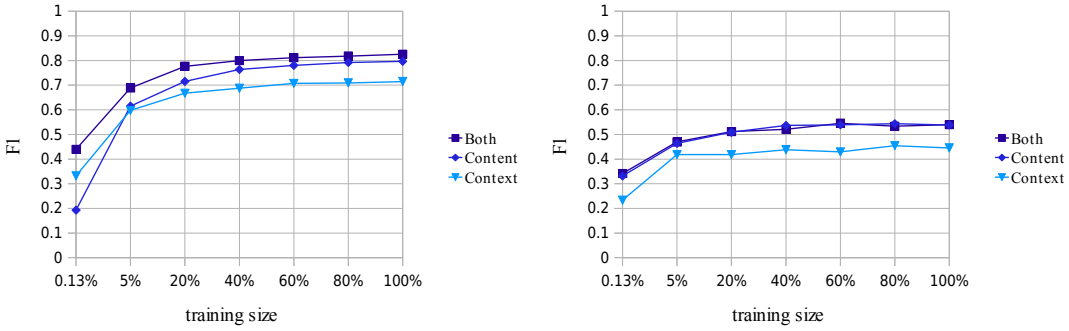


Figure 7.21: Learning curves of maximum entropy model trained on manually labeled data (a) or automatically labeled data (b).

Neighboring semantic entities as features A special type of context features are neighboring semantic entities occurring in lists or tables (e.g., Peter, Paul, an Mary). The classification of yet unknown entity candidates is embedded within an entity recognition process. Hence, it is possible using the position of recognized known entities in surrounding text. The investigation reveals the impact of neighboring entities in context features on the classification accuracy. Figure 7.22 presents the progression of accuracy values along an increasing probability that a neighboring entity is taken as context feature. The progression within the manually labeled data is as expected. Precision and recall values slightly increase for increasing probabilities. Surprisingly, the precision curves of persons and places on the automatically labeled data set decrease for increasing probabilities.

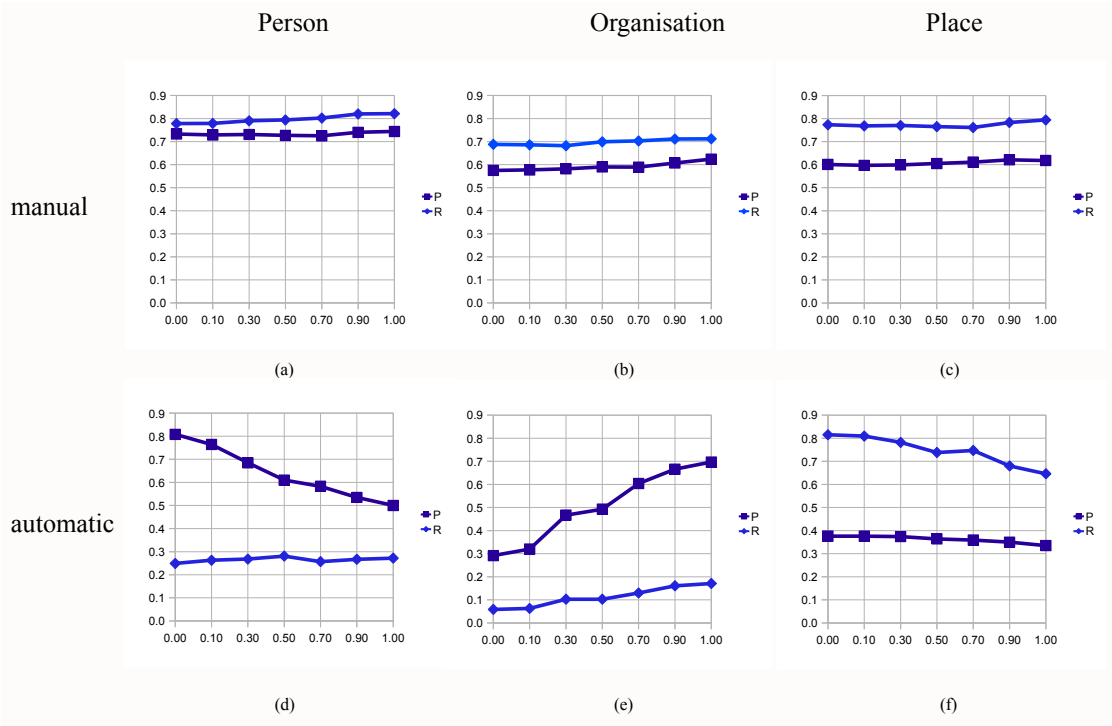


Figure 7.22: Impact of known semantic entities on context features. X-axis describes the probability that a neighboring entity is part of the feature vector.

The two confusion matrices in Tables 7.10a and 7.10b reveal that the class location suffers from over generalization. Most mistakes of misclassified examples classify these as locations. By increasing the probability of neighboring semantic entities to one, a couple of these misclassified examples are now again misclassified as persons and organizations. This also explains the decreasing recall values for places on automatically labeled data.

Again, the ambiguous usage of organizations and locations invokes problems while training a classifier on automatically labeled data.

Thresholding certainties of class predictions.

A mechanism for creating a threshold that regulates classification accuracies from either focusing on high recall or high precision values is proposed. Classification results of a maximum entropy classifier are represented as list of probabilities. Each possible classification is assigned with the probability. Based on these probabilities, the threshold is defined as the highest probability minus second-highest probability. If this subtraction exceeds the passed threshold, the predicted classification is assumed as certain enough,

sem.entity	Locat	Person	Organ
Location	1 452	50	19
Person	906	442	145
Organ	970	25	76

(a)probability=0

sem.entity	Locat	Person	Organ
Location	1 164	370	3
Person	940	437	217
Organ	853	54	186

(b)probability=1

Table 7.10: Confusion matrices listing two types of context features.

else the result is labeled as uncertain. Figure 7.23 lists precision, recall, and F-measure progressions for each feature type, and a combination of both on the manually and automatically labeled dataset along a range of threshold between 0.0 to 0.9.

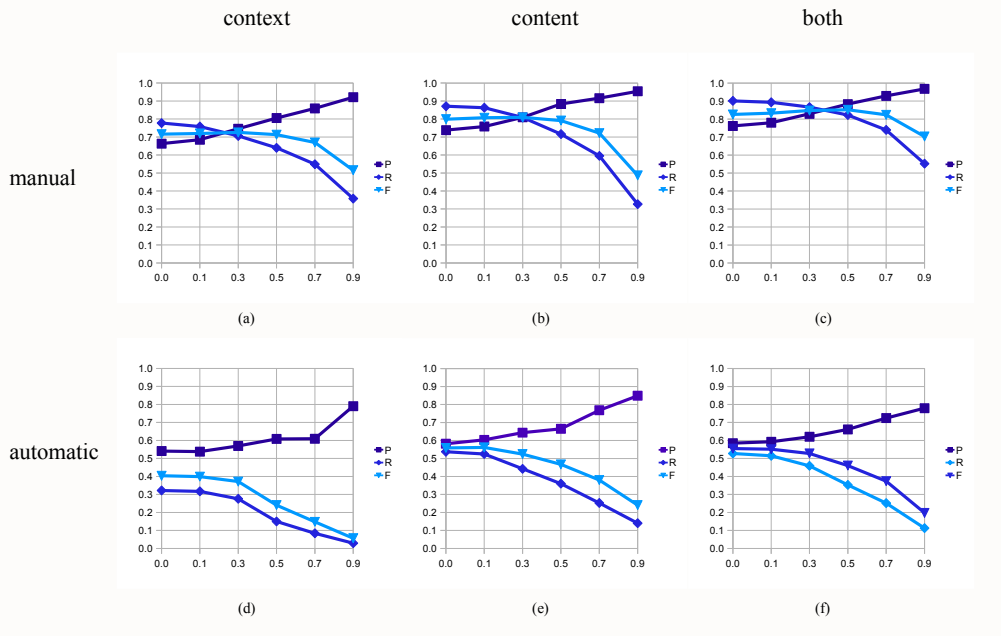


Figure 7.23: Correlation between threshold and classification accuracies.

These curves demonstrate that on the manually labeled corpus, general precision ratios may reach values above 0.9 with a recall above 0.3. Even on automatically labeled data precision values stay above 0.8. Unfortunately, the recall values fall to a range around 0.1. Hence, certainty threshold provide a means for heightening precision ratios of classifiers that were trained on automatically labeled training data.

Predicting properties and types of semantic entities

The preceding study concentrated on predicting classes such as person, organization, or location to entities. The following side study investigates the prediction of different RDF properties for similar types of entities. Therefore, the corpus provided by the Pascal Challenge on Evaluating Machine Learning for Information Extraction is used [Ireson et al., 2005]. It consists of a simple vocabulary and labeled text documents, which are call for papers taken from scientific mailing lists. Figure 7.24 shows the general ontology used to markup text passages.

classes			
Workshop		**Conference**	
name:	<string>	name:	<string>
acronym:	<string>	acronym:	<string>
homepage:	<URL>	homepage:	<URL>
general properties			
date:	<date>		
camera ready copy due:	<date>		
paper submission:	<date>		
notification of acceptance:	<date>		
person:	<string>		
location:	<string>		

Figure 7.24: Pascal ontology of workshops and conferences.

In addition to the contained entity type person, location, url, conference, and workshop, the vocabulary contains datatype properties, i.e., paper submission date, notification of acceptance date, and camera ready copy due. The Pascal corpus is used for evaluating the quality of feature types in cases where the entity class remains the same but the datatype property changes.

The same set of features was analyzed like in the original classification scenario but no significant conspicuousness could be revealed. Hence, a comparison was performed on the classifier's accuracies about either predicted properties or types of entities of the Pascal corpus. Figure 7.25 reveals that predicting properties is much more complicated than predicting types. Whereas in the case of predicted types the F-measure ratio of dates is above 0.95, the same F-measure ratios decrease to 0.8 and below for all predicted properties of type date. Interestingly, the classifier is not able to distinguish between conference and workshop homepages. Workshop homepages received higher values because of their higher probability (0.77) of occurrence in data.

In contrast to the irregularities of threshold performance curves that occurred in the ConLL2003 corpus (see Figure 7.26), the precision curves increase monotonously up to a threshold of 0.7. The precision and recall curves in Figure 7.26 indicate a positive

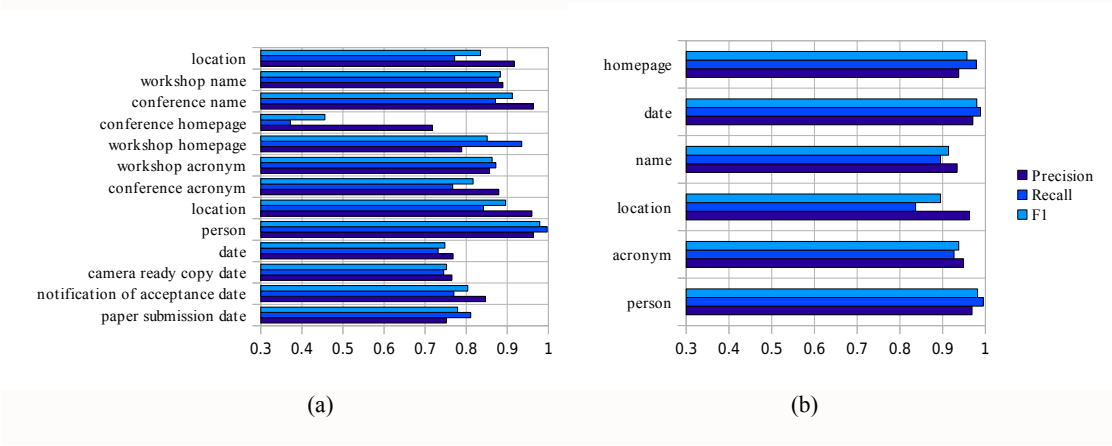


Figure 7.25: Classifier's accuracies of predicted properties (a) and types (b).

correlation between increasing the threshold and precision values.

Finally, it can be concluded that the classification experiments showed that it is possible train classifiers to automatically labeled corpus data. This allows an adaption of the IE system to a given domain of concern if it is described in RDF and an unlabeled document corpus exists for training purpose. This confirms the hypotheses H.1 and H.3.

The experiment performed on the Pascal corpus revealed that it is possible to use exactly the same classification approach for training a classifier to learn labeling recognized entities with datatype properties.

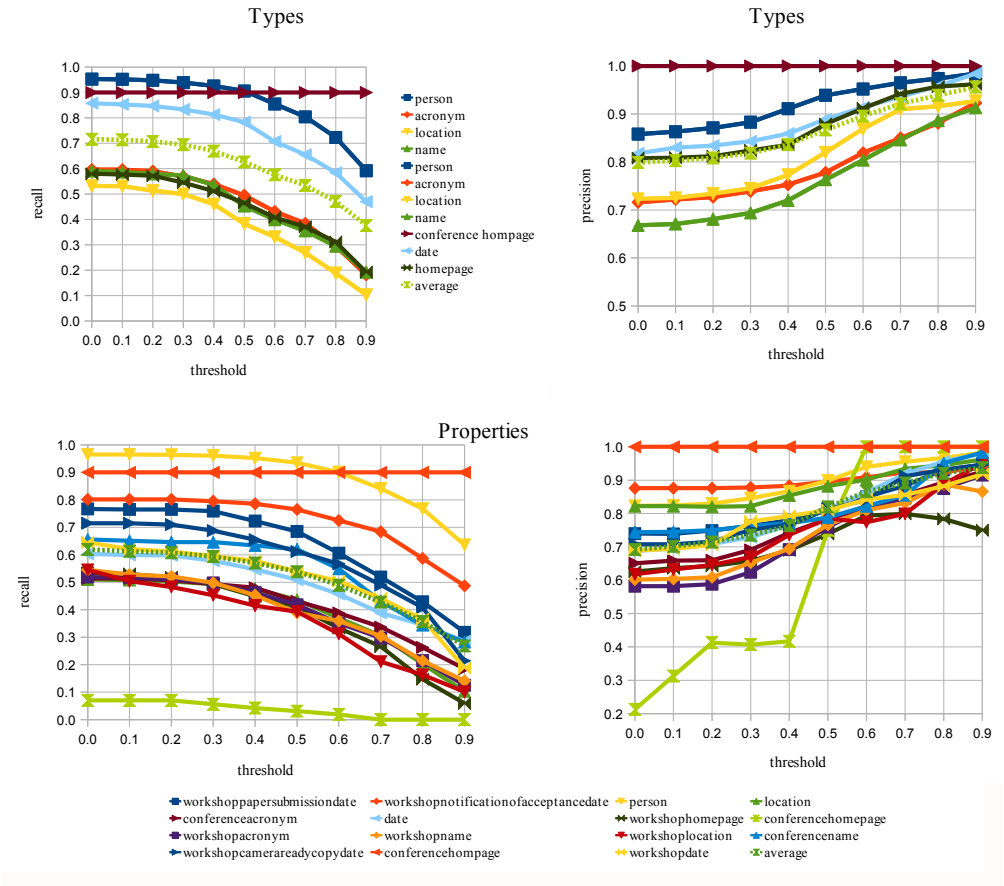


Figure 7.26: Correlation between threshold and classification accuracies.

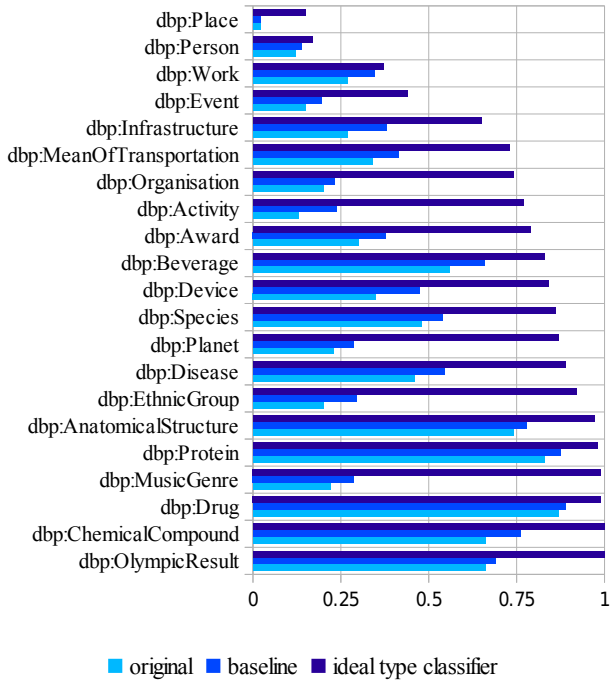


Figure 7.27: Precision ratios of an ideal type classifier on ambiguous entities.

7.9.8 Classification-based disambiguation

The goal of this investigation is to reveal the performance of an entity classification model that is applied to resolve ambiguously recognized entities. If, for example, the entity "Paris" occurs in a text, a classifier would decide based on the surrounding textual context that Paris is more likely meant to refer to a city, instead of a person. Figure 7.27 illustrates precision values of classification-based disambiguation approaches on class level. The experiment is performed on the Wikipedia corpus. For each class, the bar chart in Figure 7.27 presents precision values of three approaches:

1. The precision value of the original recognition results with an average of 0.35.

2. The precision of a baseline approach that votes for those instances whose classification is dominating the respective ambiguity set. The average is 0.4.

3. The precision of an ideal classifier that is capable to predict the correct entity type for every literal value in text. The precision values of the ideal classifier reached an average 0.78.

Interestingly, the precision values of the common entity classes such as places, persons, works, and events stay below a value of 0.5. A reason for this is that instances within ambiguity sets often share these classes. Based on the analysis of Section 7.9.7 it can be asserted that, depending on the amount of existing training, the disambiguation performance of a resulting classifier stays between the performance ratios of the baseline approach and the ideally asserted classifier. For example, training a classifier on randomly chosen and automatically labeled Wikipedia articles results in a precision ratio of 0.34 on the Wikinews corpus. It mainly consists of the common entity classes. In summary, the classification-based disambiguation is possible but not suitable to be performed on all types of classes used within an RDF graph. It can be used as additional indicator for disambiguating semantic entities.

7.9.9 Ranking extraction results by relevance

This experiment analyzes the performance of a number of rating metrics for ranking recognized semantic entities by relevance. The following metrics were investigated.

Text corpus-based metrics are computed by calculating statistics on word occurrences.

- **POS:** The position of the first occurrence in the text, as it was defined in Section 7.6.
- **TF:** The term frequency of the entity reference in the text (see Section 6.8).
- **IDF:** The overall inverse document frequency of the entity reference in the text corpus (see Section 6.9).

Graph-based metrics are computed by calculating statistics on node connectivities in RDF graphs.

- **DEG:** The node degree of incoming or outgoing edges of the entity referent in the RDF graph (see Definition 5.15).
- **CAP:** The node's capacity of incoming and outgoing edges of the entity referent in the RDF graph (see Definition 5.16).
- **PR:** Pagerank values of the entity referent in the RDF graph based on the identically named algorithm (see Definition 5.18) .
- **HUB:** Hub values of the entity referent in the RDF graph based on the HITS algorithm (see Definition 5.17) .
- **AUTH:** Authority values of the entity referent in the RDF graph based on the HITS algorithm (see Definition 5.17).

The naive baseline approach determines the most naive approach for implementing a ranking. A comparison with this baseline should indicate how well an entity ranker performs.

RANDOM: A random floating point number between zero and one. The random ranking is used as baseline in this experiment.

The experiments are conducted on all five test corpora described in Section 7.9.1. Results are rated in terms of mean average precision (*MAP*) ratios as described in Section 7.9.2. Table 7.11 lists *MAP* values for each of the described types of metrics on all five corpora. The maximal achieved *MAP* value is highlighted in bold letters.

	metric	$\mathbf{BBC}_{music}$	$\mathbf{BBC}_{nature}$	Wikinews	Wikipedia	Gutenberg
0	POS	**0.825**	**0.457**	**0.254**	0.147	**0.169**
1	TF	0.711	0.391	0.2	0.15	0.046
2	IDF	0.704	0.395	0.218	**0.158**	0.045
3	AUTH	**0.667**	0.406	0.443	0.341	0.279
4	HUB	0.339	0.504	0.409	0.244	**0.331**
5	PR	0.657	0.407	0.426	0.376	0.208
6	DEG	0.36	**0.518**	0.4	0.225	0.325
7	CAP	0.661	0.406	**0.484**	**0.38**	0.323
8	RANDOM	0.419	0.39	0.235	0.128	0.053

Table 7.11: Mean average precision ratios of ranking metrics on each corpus.

Considering text corpus-based metrics, Table 7.11 shows that the metric *IDF* and *POS* perform well for ranking instances by relevance. Hence, it can be stated that entity references to relevant instances occur frequently in beginning of text documents. If a reference to single instances occurs very frequently in a document corpus, it is most likely irrelevant to a specific information need. Values of *TF* are often nearby the random *MAP* ratings, indicating that the term frequency of entity references does not correlate with the notion of relevance.

Considering the graph based metrics, the results on both BBC corpora (music, nature) significantly differ from results on the DBpedia related corpora (Wikinews, Wikipedia, Gutenberg). Figure 7.28 allows a further investigation on this phenomenon. It illustrates the correlations coefficients between the five corpora in terms of the computed rankings. The degree of red denotes the strength of a positive correlation. The index numbers from 0 to 4 correspond with the ordering of the columns of Table 7.11 (BBC_{music}, BBC_{nature}, Wikinews, Wikipedia, Gutenberg).

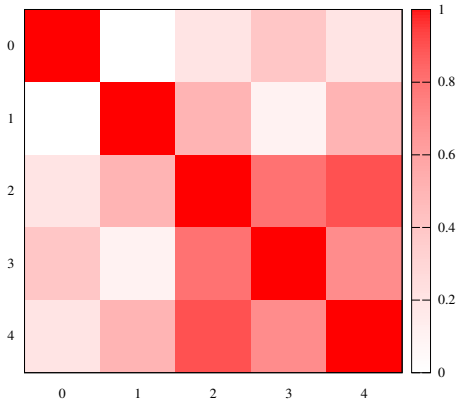


Figure 7.28: Ranking correlations between the five test corpora. 0. BBC music, 1. BBC nature, 2. Wikinews, 3. Wikipedia, and 4. Gutenberg

It can be seen that both BBC corpora do not correlate in terms of rankings. They also do not correlate to any of the DBpedia related corpora. Conversely, the DBpedia related corpora (Wikinews, Wikipedia, Gutenberg) share correlating rankings between each other. This leads to the conclusion that the underlying RDF graph has a stronger impact on the computation of ratings than the documents within the corpora have. Table 7.11 shows that DBpedia related corpora share high rated ratings computed on the basis of the node capacity (CAP) metric.

In order to combine multiple metrics for further enhancing the quality of rankings, it is helpful investigating statistical correlations between each metric. Therefore, Spearman's rank correlation coefficient, which is a specialized correlation coefficient for investigating dependencies between rankings was applied.

Definition 7.9 (*Spearman's rank correlation coefficient*)

Spearman correlations can be defined as an extension of Pearson's product momentum, which was already explained in Section 5.4.4. In addition to Pearson's product momentum, which analyzes the strength and direction of a linear dependency between two variables, Spearman's rank correlation computes the product momentum on the rank of each of the variable's values. The rank is the index of the specific value within an ordered list of all values of the respective variable. Hence, Spearman's rank correlation analyzes the monotonicity of the dependency between two variables.

Figure 7.29 illustrates heat maps on correlation matrices. Each index denotes a metric corresponding to this numbered enumeration: *0. BBC music*, *1. BBC nature*, *2. Wikinews*, *3. Wikipedia*, and *4. Gutenberg*. Again, the degree of red denotes the strength of a positive correlation. Negative correlations do not occur in this scenario, which indicates the absence of reverse rankings. The heat map of the Gutenberg corpus is equal to the heat maps of the Wikinews (Fig. 7.29c) and Wikipedia (Fig. 7.29d) corpora. Hence, it was spared adding it to Figure 7.29.

It can be observed that strong correlations exist between the rankings produced by the graph-based metrics. Above all, the metrics *AUTH*, *PR*, and *CAP* share strong correlations across all test corpora. DBpedia related corpora possess higher correlations between graph-based metrics than the others.

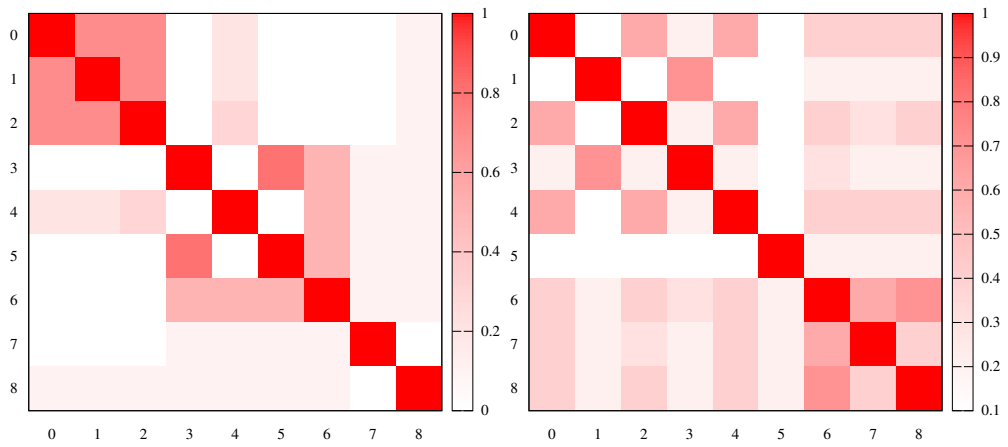
Because of the low correlations between graph-based and corpus-based metrics, it can be assumed that combining both kind of ratings results in higher rated rankings. Table 7.12 lists the top-ranked *MAP* ratings, on average and for each test corpus.

best combination	**BBC**$_{music}$	**BBC**$_{nature}$	**Wikinews**	**Wikipedia**	**Gutenberg**
AUTH·HUB·CAP·POS·IDF	0.82	0.53	0.46	0.37	0.37
AUTH·POS	**0.88**	0.46	0.44	0.36	0.29
HUB·PR·DEG·CAP·POS·IDF	0.69	**0.54**	0.41	0.30	0.38
CAP	0.66	0.41	**0.48**	0.38	0.32
PR·CAP·IDF	0.80	0.41	0.45	**0.42**	0.21
HUB·PR·DEG·CAP·POS	0.57	0.54	0.41	0.26	**0.38**

Table 7.12: Best rankings from all kinds of combinations of metrics.

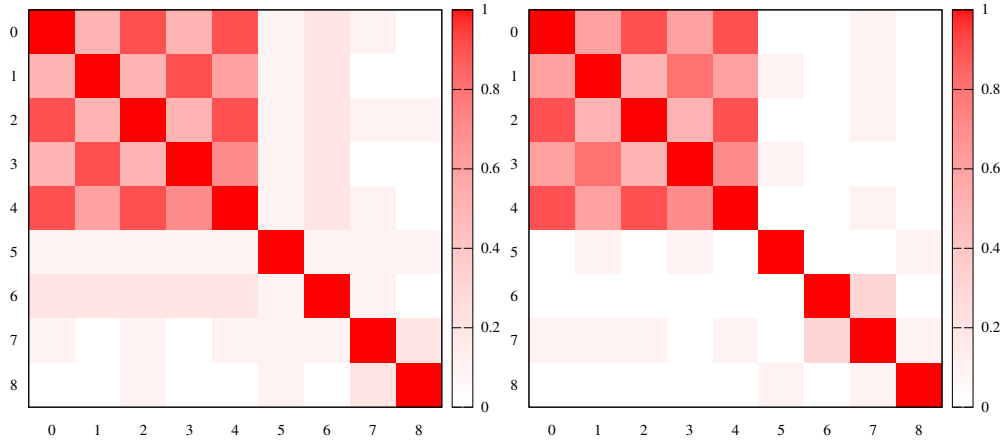
On average, the combination of metrics achieved similar or higher *MAP* ratios compared to the individual metrics. In general, *AUTH·HUB ·CAP·POS·IDF* produces the highest valued ratings. The remainder rows show the best rated ranking of each corpus. Despite the Wikinews corpus, the combination of metrics rated best combine corpus based statistics with graph based statistics. In general, the authority scores of the HIT algorithm (*AUTH*) and PageRank values (*PR*) result in similar rankings. Within the DBpedia related corpora, the node capacity (*CAP*) performs better than the combination of both HIT scores (*HUB* and *AUTH*). The corpus based metrics *POS* and *IDF* seem to enhance ratings on average.

The experiment confirms the Hypothesis H.1. Utilizing link knowledge in RDF graphs allows rating recognized instances by relevance. On average, combining knowledge from RDF graphs and text corpora increases the performance of ranking recognized entities. *AUTH·HUB·CAP·POS·IDF* work as default setting. However, on a new RDF graph the evaluation of metrics and combinations should result in an increase of *MAP* values.



(a)BBC nature

(b)BBC music



(c)Wikinews

(d)Wikipedia

Figure 7.29: Correlation heat maps between rankings:
(0. AUTH, 1. HUB, 2. PR, 3. DEG, 4. CAP, 5. RANDOM,
6. POS, 7. TF, 8. IDF).

7.9.10 Predicting object properties

This experiment investigates the possibility to predict new object properties between recognized instances by using knowledge about existing object properties. In Section 7.8 two fact prediction approaches are presented, the first using Markov chains, the second using recommender technologies.

The experimental setup is initialized as a leave one out cross evaluation. For each set of recognized instances, the set of RDF triples of these and between these instances is generated, with object properties as predicates. For a set of n RDF triples, where each respective subject and object was recognized in text, n test runs are made. Within each test run r_i with $0 < i \leq n$, the triple of index i is taken out of the set. The fact predictor is now evaluated on re-predicting exactly the taken RDF triple.

In Table 7.13, results of this cross evaluation runs are illustrated. For each evaluation run, the prediction accuracy and the overall number of predicted triples is listed. The simple Markov chain based approach (mc) is evaluated on parameter $0 < k \leq 5$ denoting the k most probable object properties between two instances of a certain class.

	accuracy			**count of predicted triples**		
approach	BBC music	BBC nature	Wikinews	BBC music	BBC nature	Wikinews
mc 1	0.56	0.93	0.46	118	253	57
mc 2	0.57	0.93	0.48	199	256	111
mc 3	0.57	0.93	0.49	228	256	156
mc 4	0.57	0.93	0.51	230	256	195
mc 5	0.57	0.93	0.51	230	256	232
mc 1 (card)	0.47	0.54	0.01	101	239	6
mc 2 (card)	0.48	0.54	0.04	182	242	59
mc 3 (card)	0.48	0.54	0.04	210	242	104
mc 4 (card)	0.48	0.54	0.07	213	242	143
mc 5 (card)	0.48	0.54	0.07	213	242	180
cf(cos)	0.41	NA	NA	26	NA	NA
cf(cor)	0.36	NA	NA	4	NA	NA

Table 7.13: Leave one out evaluation results of fact predictors.

In addition, the Markov-chain-based approach is extended by utilizing knowledge about cardinalities of object properties (see Section 6.4.2). If a given instance possesses less or equal RDF triples with an object property having average subject cardinality on the global RDF graph, the predicted RDF triple of this object property is considered as valid. Otherwise, if the given instance possesses more RDF triples of this object property

than the mean cardinality, the predicting is rejected. Finally, the recommender based approach is investigated (cf) on predicting facts by using either cosine similarities (cos) or correlation coefficients (cor).

The highest accuracy is achieved by using the simple Markov chain approach. It can be observed that an increase of k most probable object properties does not increase the accuracy significantly. Instead, the number of predicted facts increases. The cardinality restriction leads to an decrease of predicted facts, but it also decreases accuracy ratios significantly. Unfortunately, the collaborative filtering approach is based on a computational expensive correlation or cosine similarity matrix. Therefore, within the scale of minutes results could only be produces for documents within the BBC music corpus as this is the corpus with the smallest amount of object properties. Here, the achieved accuracy can not reach the performance of the Markov chain based approach, but in terms of the count of triples, it generates only a fraction of predictions.

Nevertheless, this experiment shows that the restrictive use of co-occurring instances and known links between them is sufficient for predicting reasonable object properties between not yet linked instances. This approves the Hypothesis H.1.

7.10 Summary and Conclusion

This chapter illustrated the utilization of RDF graphs for implementing information extractors, which is closely related to the Hypothesis H.1.

7.10.1 Summary

The following information extractors were covered and experimentally evaluated:

Filtering text for proper names (see Section 7.2) Experiments conducted in Section 7.9.3 approve the question if noun phrase filtering provide a scalable basis for recognizing semantic entities. The application of noun phrase chunking to the Semantic Entity Recognition involves language dependency. Hence, required technologies for applying the Semantic Entity Recognition to a specific language are:

1. The actual language must be recognized by a language identifier.
2. A word and sentence tokenizer must exist for this specific language.
3. POS-tagging on this language is required.
4. A text chunker on this language is needed.

Spotting text for datatype property values (see Section 7.3) Further experiments presented in Section 7.9.3 show that the application of prefix hashing and suffix arrays provide a scalable basis to spot for datatype properties.

Linking named entities to formal instances (see Section 7.4) In Section 7.9.4, evaluation results of a naive baseline approach recognizing semantic entities are presented.

Resolving ambiguous semantic entities (see Section 7.5) Section 7.9.6 investigates the application of link-based RDF graph metrics to resolve multiple instance references for a single recognized entity. Here, the *node degree* performed best on the test datasets (see Section 7.9.1). By nature, the presented linkage based disambiguation approaches require the existence of formal associations between instances inside the RDF graph.

In addition to using graph statistics, Section 7.9.8 evaluates the application of entity classifiers to disambiguate instances by classification. By nature, this approach should be applied in domains possessing large class hierarchies such as the DBpedia. Here, it could be revealed that the classification-based disambiguation performs best on more specialized class definitions. The probability that instances of these classes share labeling literal values is much lower than in case of more general entity classes such as organizations, persons, or locations.

Rating relevance of semantic entities in text (see Section 7.6) The application of text corpus and graphs based metric was analyzed in Section 7.9.9. Here, it could be revealed that the factor combination $AUTH{\cdot}HUB{\cdot}CAP{\cdot}POS{\cdot}IDF$ produces best rankings. Combining the information from RDF graphs and from text corpora results in an increase of performance for ranking relevance of recognized entities.

Classifying semantic entities (see Section 7.7) Section 7.9.7 investigates in detail the creation of automatically labeled corpus data to train classifiers. In terms of the DBpedia, it is possible to train a classifier on such data. However, especially the tested recall values stay significantly below the manually labeled counterparts. Furthermore, a large amount of training data is needed. Whereas the manually labeled corpus needed 219 554 words for producing 19 277 training instances (a frequency of 15 words per label), the automatically labeled corpus needed 1 804 299 words for producing 34 219 training instances (a frequency of 75 words per label). A problem of such an automatic labeling approach is that it faces problems in cases of ambiguous usage of similar literal values in different labeling contexts. This was the case in Reuter's news articles in terms of locations and organizations.

The application of maximum entropy classifiers to predict either type or properties of semantic entities showed that it is possible to train a useful machine-learning model. The overall type of analyzed features was taken from standard approaches and especially the use of text windows and n-grams resulted produced good results. The evaluation of more sophisticated features such as neighboring semantic entities revealed that the underlying classes of predicted and neighboring types have to be distinct in their values. Otherwise, the application of this feature type produces worse results. In general, it

could be observed that content-based features resulted in higher qualities and better learning curves. Nevertheless, context-based features are independent from memorized labels and thus of high importance for classifying unknown semantic entities.

Predicting object properties between semantic entities (see Section 7.8) In Section 7.9.10 two RDF graph based approaches for predicting form the perspective of an RDF graph yet unknown object properties are evaluated. Within a leave one out cross evaluation, the Markov chain based predictor produced best accuracy ratios, but also predicted the highest amount of respective RDF triples. The similarity based recommender approach has a large computational complexity. It produces the lowest accuracy ratios, but also the lowest amount of predicted triples.

7.10.2 Conclusion

The presented utilization of RDF graphs within IE resulted in contributing the following solutions listed in Section 1.3:

Contribution 2 covers the successful integration of domain information from RDF graphs into information extractors and therefore covers each of the presented information extractor in this chapter.

Contribution 3 focuses on the value of using formal vocabularies to specify the kind of information to extract from text. The experiment in Section 7.9.5 about filtering useful datatype properties provides further information on this, as datatype properties are part of the vocabulary used within an RDF graph.

Contribution 5 is dedicated to training named entity classifiers with automatically labeled training data (see Section 7.7), in which labels correspond with classes used within an RDF graph.

Contribution 6 covers the increase of adaptability by initializing an IE system with different RDF graphs describing different domains of concerns. This contribution could be created by successfully evaluating the presented information extractors on three different RDF graphs and five different document corpora (see Section 6.9.1).

8 Incorporating SPARQL and RDF serializations into Information Extraction

> . . . utilities are available to treat . . . other formats as RDF so that you can issue SPARQL queries against these data sources . . . , which is one of the most powerful aspects of the SPARQL/RDF combinations.
>
> *(DuCharme [2011], SPARQL Working Group)*

The Semantic Web intends using RDF and SPARQL for processing and filtering formal information on the Web. Representing IE results from natural language text in RDF extends this use case to unstructured content of Web pages. Following DuCharme's statement about the power of combining RDF and SPARQL, such a representation allows the filtered extraction of information from text by offering a SPARQL interface to the content of this text. By using the terms of the vocabulary of an underlying RDF graph, representing the current domain of concern, RDF based IE provides a machine-interpretable perspective on arbitrary Web pages in terms of this domain. Users can filter the content with specific bits of information by using these terms in SPARQL queries.

This chapter outlines the application of RDF, SPARQL to IE results. First, Section 8.1 introduces four post-processing operations, namely ranking, serializing, annotating, and filtering. Next, Section 8.2 provides details on ranking IE results in plain result lists. Section 8.3 describes the serialization of IE results in RDF. Based on RDF results, Section 8.4 shows the application of annotating originating text sources with IE results. Next, Section 8.5 presents the use of of SPARQL to specifying IE templates describing an information demand in terms of the underlying RDF graph. Finally, Section 8.6 summarizes and concludes the presented features.

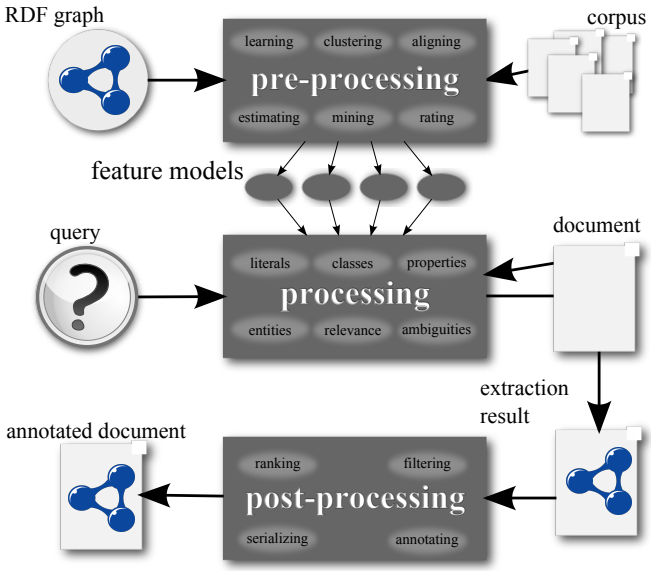


Figure 8.1: The architecture of an RDF based IE system.

8.1 Post-processing IE results

The architecture of an RDF based IE system, which is illustrated in Figure 8.1, represents IE results in RDF format.[1] A post-processing of these results offers transformations for ranking, filtering, serializing, and annotating extracted information. Following Grishman's long-term goals on IE, the presented post-processing operations enable users (humans as well as machines) to work with IE results by choosing the best suitable representation format for reaching the goals they are aiming at (see Section 2.2).

Ranking extracted information by relevance corresponds with Grishman's long-term goal of "*picking out the most useful bits*". Here, the ranking of semantic entities is implemented based on rating mechanisms (see Section 7.6). In consequence, extracted entities can be, similarly to Google search results, returned as sorted list ranked by relevance metrics. More details are given in Section 8.2.

Serializing the extracted information into syntax formats such as XML, JSON, CSV, or specializations on RDF such as TURTLE or RDF/XML means representing information machine-understandable. This corresponds directly with Grishman's demand of "*presenting extraction results in your preferred language*". Section 8.3 focusses on representing and processing IE results in RDF.

[1] See Section 5.3, where Figure 8.1 is referred to as Figure 5.7 for introducing the Semantic Entity Recognition process.

Annotating Web pages with IE results in terms of an underlying RDF graph enriches the natural language content of respective pages with formal and therefore machine-understandable hints. Section 8.4 presents an annotation of Web pages utilizing RDFa, which is a serialization format of RDF extending the HTML syntax. Hence, this post-processing is also subsumed by Grishman's demand of "*presenting extraction results in your preferred language*".

Filtering extracted information based on explicit facets (in terms of RDF properties) is a pillar of this work (see abstract of Chapter 1). Section 8.5 addresses SPARQL as aid for specifying the properties of instances the user is interested in extracting from text. The underlying intention of these filters is motivated by Grishman's demand for presenting IE results "*at the right level of detail*".

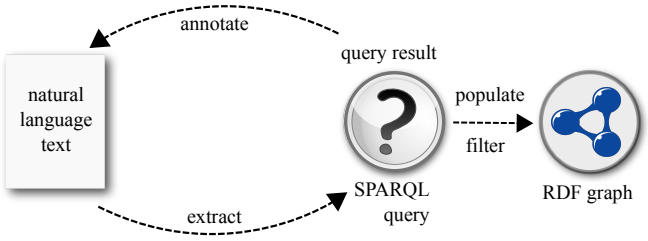


Figure 8.2: Cycle between extracting information and annotating formal knowledge.

In general, the application of RDF to representing IE results as well as the usage of SPARQL for requesting IE results confirms with the statement of the Hypothesis H.2. Figure 8.2 illustrates the impacts of the conducted investigations to the interface in between natural language text and RDF graphs. Extracting information from text based on first, an RDF graph and second, a filtering SPARQL query allows populating the underlying RDF graph with additional, yet unknown information from text. Annotating the originating text with formal information in terms of an RDF graph allows enriching the natural language text with additional information from the RDF graph. Please refer to Figure 3.6 illustrating the application of this scenario within the scope of the Semantic Web.

8.2 Ranking IE results

In Section 7.6, rating metrics are investigated for ranking recognized instances according to average precision ratings. In IR, sorted lists are used to return query results ranked by decreasing relevance values. Relevance values are computed by the relevance metrics

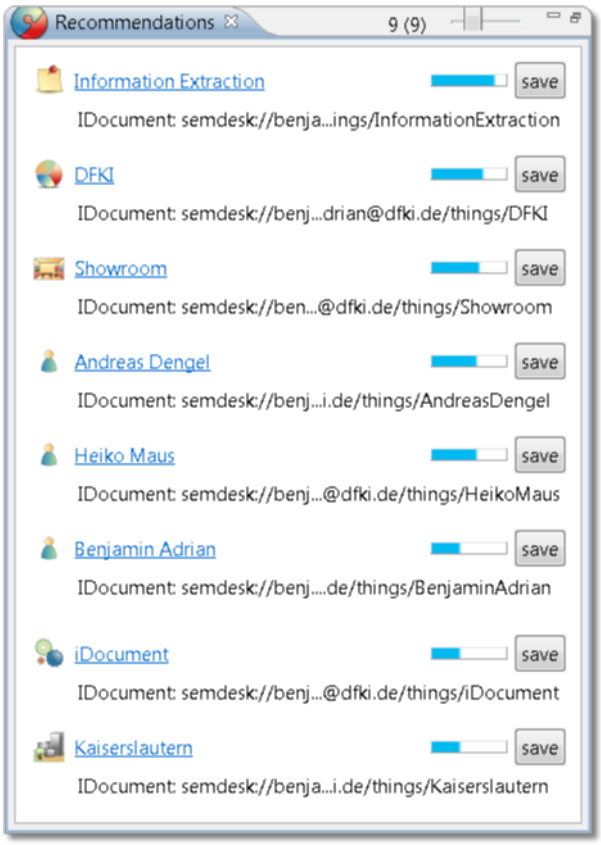


Figure 8.3: A list of recognized instances sorted by relevance.

introduces in Section 7.6. A result list sorted by relevance ratings can be paginated to hold, for example, the best-rated $k = 10$ entries.

With respect to the state-of-the-art, the most popular aim of OBIE is to recognize formal instances within natural language text (see Chapter 4). Especially when using large knowledge bases (here, RDF graphs) recognized instances in text may be rated within a range from not relevant to relevant.

Transferring the relevance ranking from IR to IE, the list-based ranking of recognized entities allows users inspecting, for example, only the most relevant k entities of a document. Figure 8.3 illustrates a screenshot of the *Nepomuk* Semantic Desktop. For a given document content, this sidebar is intended tagging documents with recognized instances of an underlying RDF graph [Adrian et al., 2008a]. In general, the user interface renders the type of instances by the use of visual icons. The label of each instance is presented

as hyperlink. Clicking on the link opens a new window with background information about the respective instance. The relevance of each recognized instance is visualized by using the horizontal blue energy bars. On the righter side, the user is able to accept each instance as tag for the given document by clicking on the button labeled with "save". The slider on the top of the window allows regulating the number of results presented to the user. Lowering the value of this slider results in filtering instances, results in rendering only a number of k best rated instances. Section 9.2.2 provides more information on this use case.

8.3 Serializing extraction results in RDF

The aim of RDF based IE is the utilization of RDF graphs as basis for extracting relevant information from text and representing the information again in RDF. For this reason, IE results have to be transformed into a graph-based format. The final output of the IE system is in an RDF graph describing recognized semantic entities, and the respective datatype and object property values. In detail, the following categories of information elements have to be covered in an RDF document representing an IE result.

The document may contain metadata such as the title, authors, date of creation and last change, the language, or its file format. Especially, the information of last change dates is important when extracting information from Web pages with evolving content, such as `http://www.slashdot.com`. Here, the Aperture framework allows extracting exactly this kinds of metadata from documents and representing it in RDF format[2].

The RDF graph underlying the RDF based IE system defines the system's current domain of concern. It defines the vocabulary, i.e., the classes, object properties, and datatype properties for describing the respective domain. The graph also lists concrete instances within this domain and describes these by the use of classifications, datatype property, and object property values. The RDF graph is identified by a URI, e.g., `http://dbpedia.org`.

Information recognized by the IE system consists of instances, which are results of the Semantic Entity Recognition. Recognized instances, their matching datatype properties, classifications, and object properties are represented as a graph identified by the following URI pattern:

Definition 8.1 (*URI of a graph representing recognized information*)

`http://[domain]?graph=[RDFgraph]&doc=[doc]#recognized`

[2] `http://aperture.sourceforge.com`

Here, the parameters `domain`, `RDFgraph`, and `doc` of this URI pattern are defined as follows:

Definition 8.2 (*URI pattern of named graphs*)

domain *denotes the the Internet domain of the deployed OBIE system. (e.g., `http://scoobie.dfki.de`).*

RDFgraph *denotes the URI of the underlying RDF graph. (e.g., `http://dbpedia.org`).*

doc *denotes the URI of the Web document. (e.g., `http://www.spiegel.de`).*

All URI strings passed as HTTP GET parameters (`graph`, `doc`) have to be URL encoded for generating a valid URI.

Predicted information consists of entities and respective properties that have not yet being instantiated by within the RDF graph. It denotes new information, yet "unknown" to the RDF graph. Predicted information is represented as RDF graph identified by the following URI pattern.

Definition 8.3 (*URI of a graph representing predicted information*)

*`http://`**[domain]**`?graph=`**[RDFgraph]**`&doc=`**[doc]**`predicted`*

In summary, an RDF based IE result comprises four graphs, describing the source document, the originating domain of concern, the recognized information, and the predicted information. Hence, RDF syntax is needed that supports the description of multiple RDF graphs within a single RDF document, e.g., the TriG syntax [Bizer and Cyganiak, 2007]. Example 8.1 illustrates the serialization of IE results in an RDF document consisting of four named RDF graphs. The example contains RDF graphs that contain statements about the document metadata, recognized instances, known information about these instances from the underlying domain, and finally novel predicted assumptions[3].

[3]Please consider that although Paul Rogers acted with Queen, he is not really a member. However, in terms of the music ontology, the prediction of the `mo:member` relationship between both instances is a good approximation of reality.

Example 8.1 (*RDF serialized IE result*)

Assuming the Web page `http://example.org` *would contain the following text.*

> *"Paul Rogers interpretation of "Queen's Born to Love You" was amazing."*

The RDF graph representing the respective IE result would be look like the following:
`http://scoobie.dfki.de?graph=http://dbpedia.org&doc=http://example.org#`

```
@prefix dc:   <http://purl.org/dc/elements/1.1/>
@prefix dbp:  <http://dbpedia.org/resource/>
@prefix mo:   <http://purl.org/ontology/mo/>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

<http://example.org> = {
 <http://example.org> dc:title ''Queen and Paul Rogers'' .
} .

:recognized = {
 dbp:Queen              a mo:MusicGroup  ; rdfs:label "Queen" .
 dbp:Paul_Rogers        a mo:MusicArtist ; rdfs:label  "Paul Rogera" .
 dbp:Born_to_love_you a dbp-ont:Work   ; dc:title "Born to Love You" .
} .

<http://dbpedia.org> = {
 dbp:Brian_May          mo:member dbp:Queen .
 dbp:Roger_Taylor       mo:member dbp:Queen .
 dbp:Freddy_Mercury     mo:member dbp:Queen .
} .

:predicted = {
 dbp:Paul_Rogers a mo:member dbp:Queen .
} .
```

8.4 Annotating IE results in Web pages

Retroactively annotating text with IE results (see Figure 8.2) allows linking recognized entities in text segments to datatype properties and respective instances within a underlying RDF graph. One can say it bridges the gap between the Web of documents and the Web of data [Adrian, 2010]. The challenge of this approach is serializing IE results within the content of the originating text.

RDFa offers a specialized syntax for embedding RDF data into attributes of HTML elements [Adida et al., 2011]. in Example 8.2, the `HTML+RDFa` snippet illustrates markup wrapping the noun phrase "Karl-Theodor zu Guttenberg".

Example 8.2 (*RDFa data in HTML elements.*)

```
<!-- HTML content extended with RDFa annotations. -->
<BODY  prefix="dbp=http://dbpedia.org/resource/
               foaf=http://xmlns.org/foaf/0.1/">  ...
  <SPAN about="dbp:Karl-Theodor_zu_Guttenberg"
        property="foaf:name">Karl-Theodor zu Guttenberg</SPAN>
```

The RDFa content annotates the phrase "Karl-Theodor zu Guttenberg" as value of a `foaf:name` of the corresponding DBpedia instance `dbp:Karl-Theodor_zu_Guttenberg`. Section 9.1.3 illustrates more details on the application of RDFa annotations.

Query 8.1 (*SPARQL template*)

```
# An IE template specified as SPARQL select query.
# Via, named graphs, it refers to the four kinds of IE results.
# Here, document metada is defined as default graph.

PREFIX obie: <http://scoobie.dfki.de?graph=http://dbpedia.org&
                                        doc=http://example.com#>
PREFIX dbp-ont: <dbp-ont:>
PREFIX rdf :    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>

SELECT *                                # the data sources
  FROM          <http://spiegel.de>     # existing document metadata
  FROM NAMED    <http://dbpedia.org>    # the underlying RDF graph
  FROM NAMED    obie:predicted          # predicted triples by the IE system
  FROM NAMED    obie:recognized         # recognized triples by the IE system

WHERE {                                 # the filter expressions
   ?doc dc:title ?title ;  dc:published ?date .
   GRAPH obie:predicted { ?s1 rdfs:label ?name1 .}
   GRAPH obie:recognized { ?s2 rdfs:label ?name2 .}
   GRAPH <http://dbpedia.org> { ?s1 rdf:type dbp-ont:Person .
                                ?s2 rdf:type dbp-ont:Person .}
}
```

8.5 Filtering IE results by specifying templates in SPARQL

Based on the RDF representation of IE results, SPARQL queries can be specified defining the properties of needed information (see Section 3.8). This is related to the Hypothesis H.2 and was proposed in [Adrian et al., 2009d]. Query 8.1 illustrates a SPARQL SELECT statement [SPARQL Working Group, 2008].

Here, the RDF graph representing document metadata is bounded as default graph. Hence, it is possible to specify selection patterns outside a graph scope. The graphs covering recognized, predicted information form the document as well as known information from the underlying RDF graph are declared as named graphs. In consequence, it is necessary to specify a respective selection pattern within the scope of the respective graph URI.

By referring to multiple URIs identifying multiple documents and respective extraction results, it is even possible to span queries across multiple documents and knowledge bases. SPARQL is designed for querying RDF graphs that can be HTTP requested by resolving the identifying URIs. Hence, by publishing IE results under specific URIs allows to process them by SPARQL [DuCharme, 2011].

In terms of the DBpedia, Mendes et al. [2011] also propose the filtering of IE results by using SPARQL queries. They focus on specifying the kind of information items that should be recognized in text. In addition to a simple filtering, the following approaches show how the RDF graph in combination with the SPARQL query can enhance the extraction process, directly.

8.5.1 Extracting Filtering Statements from SPARQL templates

Besides the filtering of extraction results, SPARQL queries can be passed directly to an RDF based IE system. Knowledge about these filtering patterns can be utilized for reducing the amount of computations for generating an extraction result. With respect to the Query 8.1, especially the filter expressions addressing recognized information and the filtering expressions covering the underlying RDF graph are used for enhancing the Semantic Entity Recognition process.

- `GRAPH obie:recognized { ...}` allows a direct filtering of recognized instances by specifying classes and datatype properties.

- `GRAPH <http://dbpedia.org> { ...}` allows a filtering of recognized instances and facts about these by specifying additional restrictions on respective properties.

Information about filtering datatype properties and classifications are suitable for reducing the number of necessary string comparisons when spotting text for datatype property values (see Section 7.3). Especially passing the set of relevant datatype properties decreases the number of candidate literals from the RDF graph. Reflecting the SQL Query 7.1 below, the set of datatype properties is already part of the candidate producing database lookup.

Query 8.2 (*Request datatype property values.*)

```
-- The filter expressions of this SQL query within the implementation of
-- the named recognition can be restricted by passing the datatype
-- properties specified in the SPARQL template.

SELECT DISTINCT L.literal, L.index, S.predicate
FROM index_literals L, symbols S
WHERE ( S.predicate IN (
                -- <list of datatype properties>
) AND S.object = L.index AND L.prefix IN (
                -- <list of hashed prefix values>
 )) ORDER BY L.literal
```

8.5.2 Inferring Filtering Statements from SPARQL templates

The existence of an explicit RDF graph consisting of both vocabulary and instances allows an entailment of manually created filtering expressions in SPARQL templates by inferring additional classifications or properties. This supports cases, in which users do not provide any explicit information about datatype properties in SPARQL templates. Instead, they just specify a list of classes the recognized instances should possess. However, the statistics about proper names of datatype properties' values, which are described in Section 6.6, are suitable for inferring a set of relevant datatype properties. Query 8.3 takes the cluster of a class (see Section 6.3) and a lower threshold of the proper name rating as input an returns a list of datatype properties that a re suitable for spotting literal values of instances of the respective class.

Query 8.3 (*Inferring datatype properties for known instance classification.*)

```
--Filtering datatype properties with high proper name ratings.

SELECT property, rating
FROM proper_noun_rating
WHERE (cluster = ? AND rating > ?)
```

For example, filtering IE results in terms of the DBpedia by `?s a dbp-ont:Person` results in the inference of relevant datatype properties: `rdfs:label` and `foaf:name` when using a threshold $rating > 0.1$. Increasing this threshold on ratings removes `foaf:name` from the results.

Even in cases where only a desired object property is referred to within the SPARQL template's filter expressions is is possible to use the Markov chain described in Section 6.4.3 for estimating classes of instances being most likely connected by the object property, respectively. For example, The Query 8.4 takes an object property and a lower

threshold on the probability as input and returns domains and ranges of the given object property.

Query 8.4 (*Inferring classes from signatures of known object properties.*)

```
--Mining types within signatures of object properties.

SELECT subject AS domain, object AS range, probability
FROM markov_chain
WHERE (predicate = ? AND probability > ?)
```

For example, specifying `?s dbp-ont:birthplace ?o` results in the inference of relevant clusters: `owl:Thing`, `dbp-ont:Place`, `dbp-ont:Cleric`, `dbp-ont:Person`, `dbp-ont:Athlete`, `dbp-ont:PopulatedPlace`, `dbp-ont:Politician`, and `dbp-ont:Artist`. Based on these classes the entailment of datatype properties infers `rdfs:label` as relevant. Here, a threshold on the average Markov probability of $probability > 0.01$ was taken.

8.6 Summary and Conclusion

Contributing to Hypothesis H.2, this chapter described the integration of RDF and SPARQL into the input/output-interfaces of an RDF-based IE system.

8.6.1 Summary

This chapter exposed the benefits of representing IE results in RDF and querying these with SPARQL.

- Section 8.1 illustrated general features of post-processing IE results, such as ranking, serializing, annotating, and filtering.
- Ranking IE results by relevance in form of lists is shown in Section 8.2. This enables users filtering the top k results or paginate the while result list in user interfaces.
- Section 8.3 described how to serialize IE results in RDF based on named graphs. Each graph represents information, which was recognized in text, predicted based on the text content, known in the RDF graph, and formally known within the Web document.
- The reverse annotation of natural language text with IE results was shown in Section 8.4 by applying an RDFa serialization.
- Providing RDF results as named graphs allows the application of SPARQL queries. Filtering expressions can span multiple extraction results and external RDF graphs

describing several domains of concerns. Besides this feature for expressing even complex information demands, the specific filter expressions are used to enhance the efficiency of matching candidate RDF literals with the document's text segments. The combined use of an RDF graph and a SPARQL query using the respectively contained vocabulary for describing instances provides the application of inference mechanisms entailing additional restrictions on classes and datatype properties.

8.6.2 Conclusion

Finally, these emerging possibilities of utilizing RDF and SPARQL conforms the Hypothesis H.2. The following contributions result from the presented integration and processing of RDF serializations and SPARQL queries:

Contributions C.3 SPARQL is shown to be suitable for specifying IE templates. More than that, it can enhance the extraction process as such. Due to the proposed inference techniques, users may underspecify SPARQL queries to a certain degree. The system is able to infer needed classifications and resulting datatype properties.

Contribution C.4 The serialization of IE results in RDF syntax and especially the reverse integration of IE results into Web pages by using RDFa enables machines to interpret natural language text in terms of an underlying RDF graph.

9 Applications of RDF-based Information Extraction

> Who sees further a dwarf or a giant? Surely a giant for his eyes are situated at a higher level than those of the dwarf. But if the dwarf is placed on the shoulders of the giant who sees further?
>
> *(Isaiah di Trani, c. 1200)*

The presented approach of RDF based IE is implemented based on Semantic Web technologies RDF and SPARQL. Here, RDF based IE means extending traditional IE techniques with features from RDF and SPARQL. This provides IE applications with a potential to utilize and filter the information about a domain of concern from a respective RDF graph. Finally, on the shoulders of Semantic Web technologies, a number of information extraction and filtering applications are implemented. They implement use cases with a support for filtering, categorizing, and enriching natural language text with information from RDF graphs.

This chapter outlines IE software systems and prototypes, which were developed based on RDF-based IE. First, Section 9.1 describes the RDF-based IE system SCOOBIE and derived applications. Second, Section 9.2 address additional application and experiments that base on results of this work. Finally, Section 9.3 summarizes the presented applications and addresses a position of their impact on Web users.

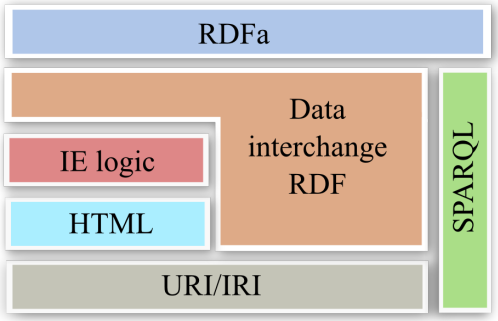


Figure 9.1: RDF-based IE-layer cake. Extraction logic is deployed as middleware component that provides an RDF view on HTML data.

9.1 RDF-based Information Extraction Systems

As already mentioned in Section 3.10, the goal of implementing an RDF-based IE system is deploying it within the technology stack of the Semantic Web. Figure 9.1 illustrates this integration in terms of IE logic, providing RDF views on HTML content (the figure is already printed on page 202). In detail, by using such an IE system, machines on the Web are enabled to interpret the natural language content (or at least parts of it) of a Web page in formal RDF terms related to an underlying RDF graph describing the actual domain of concern.

9.1.1 SCOOBIE

The approaches presented in Chapters 5, 6, 7, and 8 were implemented and evaluated based on the developed SCOOBIE prototype. At a glance, SCOOBIE is realized as follows:

- *SCOOBIE* is developed in the Java programming language.
- *SCOOBIE* implements the architecture illustrated in Figure 5.7.
- *SCOOBIE* is based on a pipeline implementation of a finite state cascade consisting of finite state transducer implementations related to each IE task.
- *SCOOBIE* stores RDF graphs in PostgresQL (see Section 6.2).
- The following Open Source libraries were used:

Mallet is a machine-learning library implementing the CRF and maximum entropy models [McCallum, 2002] described in Section 5.4.8 and 5.4.7.

Colt is a library implementing numeric algorithms on matrices[1]. Colt based matrix operations are used for calculating covariance matrices (Section 5.4.4) and implementing the hierarchical clustering (Section 5.4.3), the PCA (Section 5.4.5), and finally the similarity-based recommendations of facts (Section 7.8).

Jung is a graph implementation providing link analyzes algorithms.[2] The Jung graph implementation is used for computing graph metrics such as HITS (Section 5.4.6), or PageRank (Section 5.4.6).

Sesame is an RDF library for handling RDF graphs and SPARQL.[3] Sesame was used to parse RDF graphs (Chapter 6) and SPARQL queries (Section 8.5) and to serialize IE results into RDF (Section 8.3).

OpenNLP is a library providing text chunkers and POS taggers for English and German.[4] The text segmentation into sentences (Section 2.2.2) and POS tagging (Section 2.2.2) was performed by using OpenNLP implementations.

Nutch is an IR library providing an implementation of a language detection.[5] This features was taken as basis for identifying the language of an incoming text (Section 2.2.2).

PostgresQL is a relational database server.[6] The PostgresQL database was used for storing the domain describing RDF graph and for performing the SQL queries described in Chapters 6 and 7.

The following Java code is based on the SCOOBIE library and illustrates a typical IE program. In-line comments refer to related sections within this work.

[1] http://acs.lbl.gov/software/colt/
[2] http://jung.sourceforge.net/
[3] http://www.openrdf.org/doc/sesame2/system/
[4] http://nutch.apache.org/
[5] http://nutch.apache.org/
[6] http://www.postgresql.org/

Algorithm 9.1 (*Java source code executing SCOOBIE*)

```
// initializing the underlying RDF graph
KnowledgeBase kb = new PostgresKB( // see Section 6.2
  connection,                      // connection to postgres server
  dbName,                          // database containing the RDF graph
  "http://dbpedia.org"             // URI of the RDF graph
);

// creating the Semantic Entity Recognition pipeline
Pipeline pipe = new Pipeline(kb);  // see Section 5.3
pipe.configure(
  languageClassification, // see Section 2.2.2.2
  wordTokenizer,          // see Section 2.2.2.3
  sentenceTokenizer,      // see Section 2.2.2.3
  posTagger,              // see Section 2.2.2.4
  nounPhraseChunker,      // see Section 7.2
  suffixArrayBuilder,     // see Section 5.4.1
  namedEntityRecognizer,  // see Section 7.3
  regexEntityRecognizer,  // see Section 6.7
  namedEntityClassifier,  // see Section 7.7
  instanceResolver,       // see Section 7.4
  instanceDisambiguator,  // see Section 7.5
  factRetrieval,          // see Section 3.3
  instanceRater,          // see Section 7.6
  factRecommender         // see Section 7.8
);

// specifying the IE template in SPARQL
String template = "SELECT * FROM ... WHERE { ... }"; // see Section 8.5

// creating an internal document representation of a Web page
Document doc = pipe.createDocument("http:// ... ",template);

// execute pipeline
for (int step = 0; pipe.hasNext(step); step = pipe.execute(step, doc)) {
 log("finished "+ pipe.getTranducer(step));
}

// serialize results in turtle
TurtleSerializer turtle = new TurtleSerializer()  // see Section 8.3
Reader rdf = turtle.serialize(doc, kb);
```

SCOOBIE is published as Open Source under the Lesser Gnu Public License (LGPL) license. The source code is hosted at https://github.com/benjamin-adrian/scoobie.

9.1.2 A comparative view on Semantic Entity Recognition systems

Referring to the related work in Section 4.2, the IE systems *Zemanta*, *DBpedia Spotlight*, *Open Calais*, as well as a naive baseline implementation called *DynaQ* are compared to SCOOBIE by requesting them for recognizing instances from the DBpedia in the *Wikinews* corpus. The intention of this evaluation is to show how the generic approaches of the RDF-based IE system SCOOBIE performs compared to specialized implementations. *Zemanta* and *DBpedia Spotlight*[7] are instance recognizing systems specialized on the DBpedia RDF graph. *Open Calais* is specialized on its own proprietary RDF graph, which contains `owl:sameAs` links to instances within the DBpedia.

DynaQ is an IR engine [Reuschling et al., 2010]. It computes the top 10 rated keywords from a given text by using the *TF-IDF* metric. These keywords are used to query a *TF-IDF*-based vector space model indexing the English Wikipedia. Resulting Wikipedia articles are resolved with corresponding DBpedia instances, which are returned as recognized instances. The DynaQ-based approach is designed as a general baseline implementation of recognizing instances.

SCOOBIE results are evaluated in different settings. **SCOOBIE**$_{BL}$ denotes the baseline line implementation corresponding to Section 7.9.4. **SCOOBIE** denotes a standard configuration, in which **degree** is used for disambiguating instances. **SCOOBIE**$_{TOP10}$ denotes a standard configuration, in which the top 10 rated results according to a *capacity* based relevance rating are returned (see Section 7.9.9).

Table 9.1 illustrates the comparison results in terms of precision, recall, and F1-measure.

RDF-based IE system	precision	recall	F1
Zemanta	0.39	0.38	**0.38**
DBpedia Spotlight	0.12	**0.55**	0.20
Open Calais	0.26	0.09	0.11
SCOOBIE$_{TOP10}$	**0.42**	0.53	0.46
SCOOBIE	0.23	0.68	0.31
SCOOBIE$_{BL}$	0.01	**0.74**	0.02
DynaQ	0.03	0.07	0.04

Table 9.1: A system comparison of instance recognition services.

The Wikinews corpus consists of 100 News stories from Wikinews, which were labeled manually by six annotators. In general, it can be seen that, Wikinews is a difficult

[7] DBpedia Spotlight was configured with the parameters support=30 and confidence=0.2

corpus for performing instance recognition. All systems produced results rated with F1-measures below 0.5. *DynaQ* is confirmed as baseline, which resulted in the lowest F1-rating of 0.04.

The $SCOOBIE_{BL}$ configuration resulted in best-rated recall and worst rated precision ratings. Changing the configuration to **SCOOBIE** heightens precision but lowers recall ratios. This configuration is already competitive to the *DBpedia Spotlight* system. When restricting to the top 10 rated instances in the $\textbf{SCOOBIE}_{TOP10}$ configuration, an additional increase of precision can be achieved. It also reduces the recall ratio by 15. However, the system reaches a comparable performance like *Zemanta*.

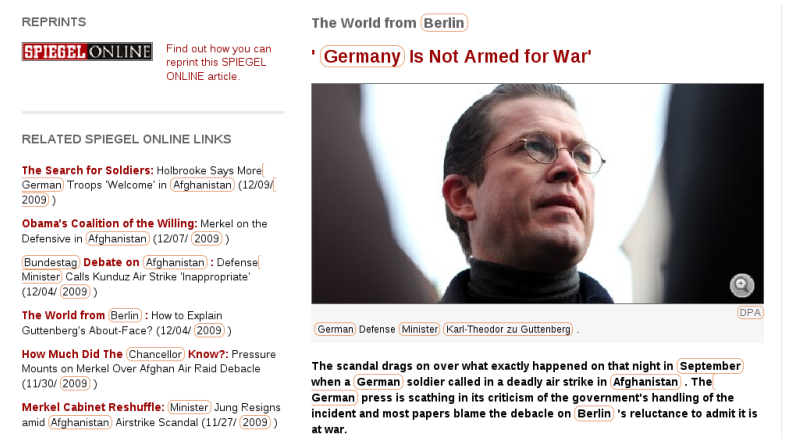
Listing some particular issues of the DBpedia RDF graph, figures out that performance can be increased by fixing ambiguities within the RDF graph.

- Multiple URIs exist for exactly the same instance. For example, `dbp:Barack_obama` and `dbp:barack_Obama` exist containing the literal values "Barack obama" and "barack Obama". Both instances link to `dbp:Barack_Obama` by `owl:sameAs`.
- Categories exist about instances. Both share the same literal values. For example, `dbp:Category:Barack_Obama` and `dbp:Barack_Obama` exist and both share the literal "Barack Obama".
- DBpedia instances exist, which correspond with disambiguation pages of the Wikipedia linking to possible instance referents. However these disambiguation instances possess the same literal values. For example, `dbp:Queen` is such a disambiguation instance possessing the literal value "Queen".

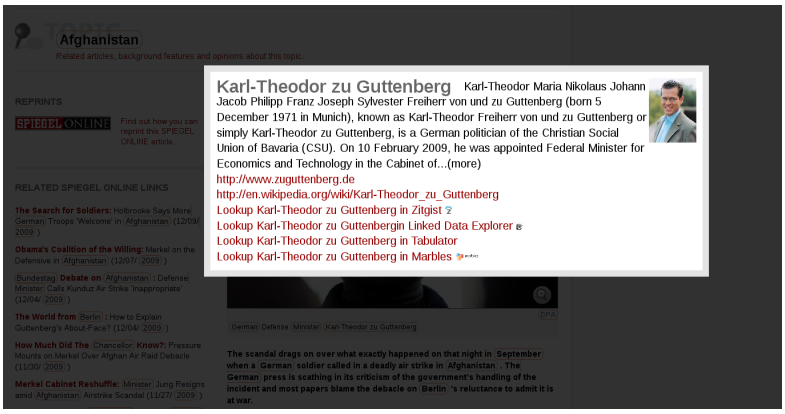
These three natures of the DBpedia produce noisy links and ambiguity groups, which lead to error prone decisions by the proposed link-based disambiguation methods. Cleaning the RDF graph from those noisy instances would most likely enhance the precision of results of at least the SCOOBIE system.

9.1.3 Epiphany

Adrian et al. [2010] propose the *Epiphany* service, which creates an RDFa serialization of IE results within the original content of a Web page. Hence, it generates a new version of a passed Web page consisting of additional RDFa markup. This allows creating multiple perspectives on a Web page's content relating to the domain of concern of the underlying RDF graph [Adrian and Dengel, 2011]. *Epiphany*'s generated RDFa markup integrates metadata directly into text passages of Web pages and thus annotates and thereby links these text passages as logical topics of the document to additional information provided by external LOD datasets. *Epiphany* adds stylesheet information to the RDFa enhanced Web page highlighting RDFa content with colored borders (see screenshot in Fig. 9.2a).



(a)Highlighted RDFa markup in a Web page.



(b)Requesting instance information about from a LOD dataset.

Figure 9.2: The Epiphany service linking instances from LOD graphs to semantic entities via RDFa.

In addition, a mouse click on an RDFa enriched phrases triggers an asynchronous HTTP request to the URI address of the respective instance returning more information about the semantic entity's referent in RDF. Finally, the resulting RDF information is rendered into an HTML template. Figure 9.2b shows such a rendered information result in style of a light box.

Epiphany figures out that RDF based IE allows enriching Web pages content with additional information on recognized instances from an RDF graph. It is required that the RDF graph is published as LOD, which means each URI of a contained instance can be requested for RDF information via HTTP.

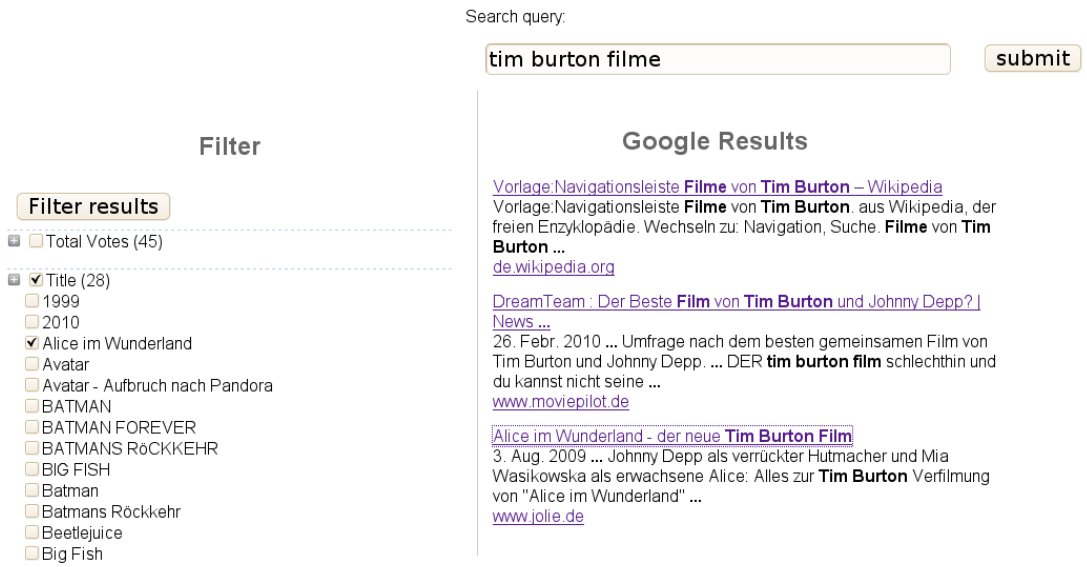


Figure 9.3: Screenshot of the Sterntaler faceted Web search.

9.1.4 Sterntaler

The *Epiphany* application consumed IE results in RDF within the scope of a single Web page. In addition, the implemented Sterntaler faceted search application aggregates and utilizes RDF results from multiple Web documents. The goal of Sterntaler is to collect facets in terms of properties of recognized instances. These facets are used to populate of a set of typed filters. Sterntaler integrates into a Web search engine. The developed prototype was implemented based on the Google API. For a returned search result list of documents, Sterntaler aggregates the RDF graphs extracted by SCOOBIE into a mashup underlying the complete search result. An analysis of instances' properties (e.g., prices, ratings, names) and their respective values returns frequently used properties that Sterntaler uses for rendering faceted filters. Here, each property's values populates its own filter.

Figure 9.3 shows the user interface of Sterntaler that used SCOOBIE on a LOD dataset about products by Amazon.com. Extracted filters are about product prizes, product ratings, product titles, and overall ratings. The user is now enabled to select certain filter values. This causes Sterntaler to present only those documents in the result list whose scenario graphs contain instances that have known properties and values corresponding to the selected filters and values.

The Sterntaler application illustrates the impact of utilizing LOD in form of RDF graphs and IE for enhancing the IR functionality of Web search engines. Based on

the existing information from the RDF graph, properties and respective values from recognized instances are aggregated. This provides users with an interface for filtering the returned list of documents in terms of structured facets. Here, the contribution of RDF based IE is the automatic extraction of these facets in terms of properties of recognized instances.

9.2 Additional applications and experiments

Based on this research, a number of additional applications and experiments were conducted. Each application is implemented by using SCOOBIE either as software library component or as service.

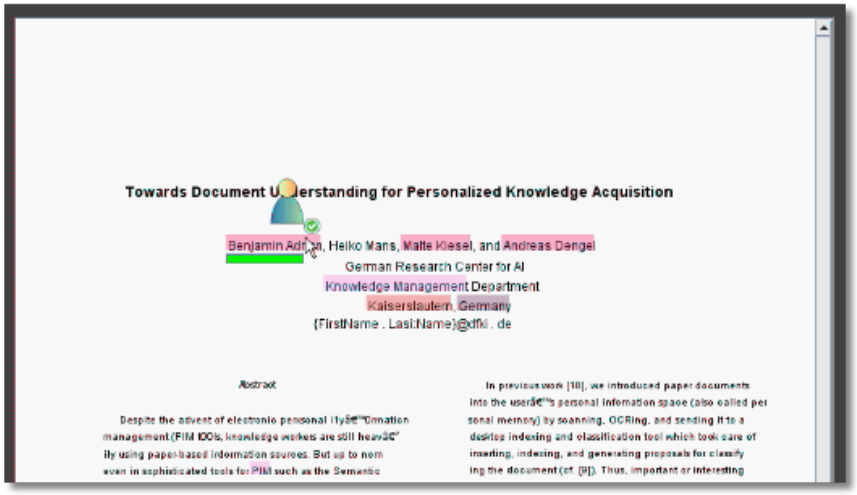


Figure 9.4: Rendering recognized semantic entities on top of a document image.

9.2.1 Labeling a digital document image

Although the focus of this work is set on extracting information from Web documents, Adrian et al. [2009e] describe an additional experiment conducted on scanned documents. Based on a scanned document, an Optical Character Recognition (OCR) was performed [Breuel, 2008]. This results in a plain text extraction contained in the document. SCOOBIE processed this plain text by using an RDF graph from a Semantic Desktop system (see Section 9.2.2). On top of the digitalized document image, a transparent layer was placed. On this layer, a visual projection of the IE results was rendered. Figure 9.4 illustrates parts of a screenshot of the developed user interface. It highlights

recognized entities. For a recognized entity in text, which is hovered by the mouse on the respective document image, the user interface provides an image icon representing the entity's class, and the relevance ratio of the recognized entity in form of a vertical bar. The user is enabled to acknowledge each recognized entity in order to categorize the document with the respective instance referent of the underlying RDF graph. A similar scenario was performed by Ebert et al. [2010] on handwritten text; this approach was also implemented on the basis of SCOOBIE.

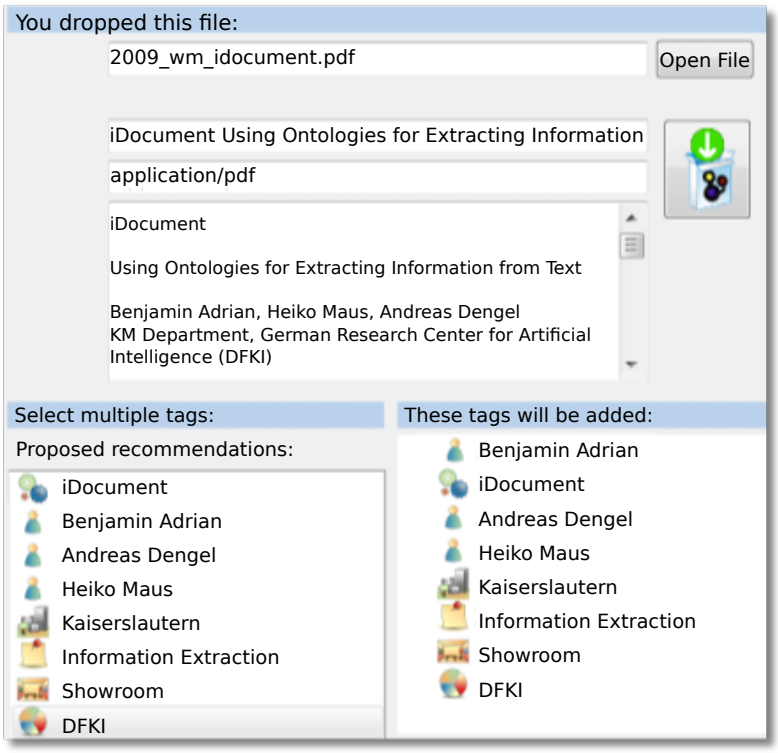
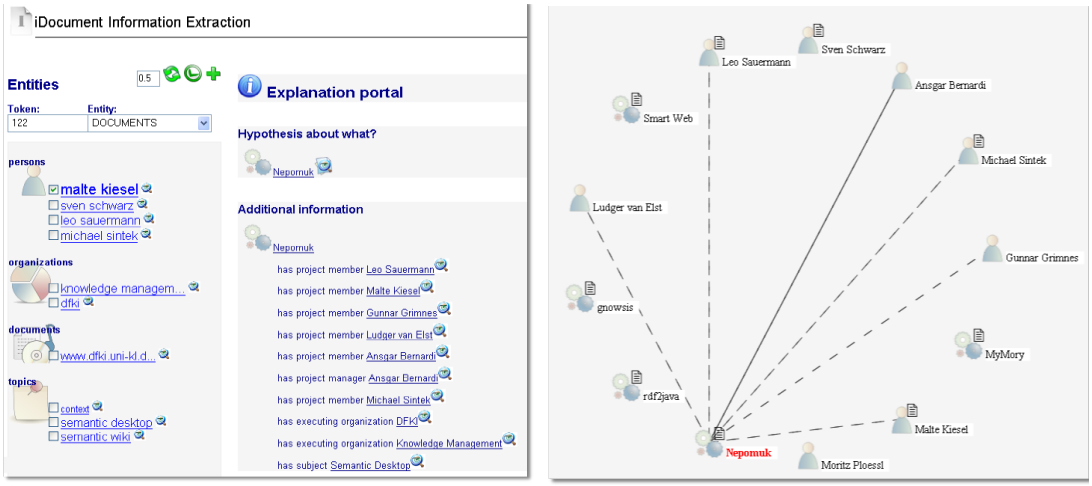


Figure 9.5: The Nepomuk DropBox providing recognized instances as tag recommendations.

9.2.2 Semantic Desktop

The idea of a Semantic Desktop comprises an RDF graph describing a personal information model of a single user [Sauermann et al., 2006]. Within the Gnowsis and Nepomuk implementations of a Semantic Desktop, Dengel and Adrian [2011] applied SCOOBIE for recognizing instances representing personal concepts of the respective user in text.

Figure 9.5 illustrates a screenshot of the Nepomuk DropBox [Groza et al., 2007]. Here, the user receives recognized instances from a text document as categorizing tag recommendations. The upper part of the window lists document metadata, extracted on the basis of the Aperture library. The lower left quarter contains a list of tag recommendations. The lower rightern quarter contains a list of tags, which are approved by the user. The DropBox is associated with a folder in the file system. Whenever the user drops a document into this folder, the DropBox appears. In a user study, Adrian et al. [2009c] evaluated the quality of the Nepomuk internal instance recommender compared to SCOOBIE results. Both systems produce different tag recommendations, which were rated comparable by the users.



(a)Explanations about the recognized instance "Nepomuk".

(b)Explanations about "Nepomuk" in form of a graph.

Figure 9.6: (a)Tabular-based and (b) graph-based explanations on IE results.

9.2.3 Explanations

Handling formal RDF data within intermediate results in the OBIE pipeline, allows generating formal explanations on why, how, what, and where a semantic entity was extracted. Forcher et al. [2008b] used iDocument, which was an early version of SCOOBIE, for integrating general explanations on IE results in a specialized user interface. For a given text, Figure 9.6 illustrates background information from an RDF graph about a recognized instance in form of explanations. Figure 9.6a shows on the left side a list of recognized instances grouped by types. This user is able to accept instances as categories by using the checkboxes. Besides each instance is an icon in form of a magnifier.

When clicking on this icon, an explanation about this instance appears (here, about the instance "Nepomuk"). In this visualization, the explanation on the right side of Figure 9.6a provides background information from the RDF graph about the respective instance. Figure 9.6b shows a graph-based rendering of this explanation. A document symbol attached to an icon denotes that this instance is recognized in text. Forcher et al. [2008a] proposed a generalized approach an integrating explanation aware computing illustrating iDocument as showcase. Adrian et al. [2009a] continued the activities on explanations by investigating a user study about the impact of explanations on rating relevance of IE results within a Semantic Desktop environment.

9.2.4 Textual case-based reasoning

Textual case-based reasoning investigates transforming textual descriptions on solving a problem (e.g., cooking a meal by using a recipe) into a formal case representation. Such a case can be retrieved, reused, and adapted to similar problems. Roth-Berghofer and Adrian [2010] developed a system based on SCOOBIE, which uses an underlying RDF graph representing a case base, for transforming textual case descriptions to formal RDF representations. Roth-Berghofer et al. [2010] proposed using the content enrichment based on SCOOBIE for enhancing text-based case description with additional information from LOD sources.

9.3 Summary and Conclusion

The applications presented in this chapter utilize a version of the RDF based IE SCOOBIE in form of a software library or service. SCOOBIE allows processing and interpreting the natural language content of documents in terms of an underlying RDF graph. The extracted results follow the formal semantics of the underlying RDF graphs. Hence, processing the document content in form of the IE results in RDF becomes as simple as writing an ordinary Semantic Web application.

9.3.1 Semantic content enrichment

Section 9.1 introduces the SCOOBIE system and compares it with related work. SCOOBIE is an open source reference implementation of the presented RDF based IE approach. Next, Section 9.2 lists additional applications and experiments that relate to directly SCOOBIE. The presented applications use and process IE results in similar ways.

- Epiphany automatically generates RDFa annotations for enriching text content with additional information from RDF graphs. (see Section 9.1.3)
- Sterntaler provides an automated extraction of semantic facets for navigating and filtering within a Web search result. (see Section 9.1.4)

- The Nepomuk DropBox supports users in categorizing documents within their personal information model by proposing category recommendations.(see Section 9.2.2)

In general, the RDF based IE solution enables applications to recognize instances from RDF graphs in natural language text and represent the recognition results again in RDF. Enriching the respective semantic entities with additional information from the underlying RDF graph results in the following process for content enrichment.

Example 9.1 (*Content enrichment*)

This content enrichment process utilizes semantic entities for generating a semantic mashup by combining the natural language text content with RDF content from the RDF graph.

1. *The enrichment process starts with analyzing text passages in original Web pages from the perspective of a dedicated domain of concern.*

   "From Port Angeles I carried on towards Forks on highway 101."

2. *The RDF graph describing the domain of concern is utilized by the RDF-based IE system to recognize semantic entities in text and annotate these with RDFa markup.*

```
From <a about="dbp:Port Angeles"
        property="foaf:name">Port Angeles</a>
    I carried on towards
<a about="dbp:Folks%2C Washington"
    property="foaf:name">Forks</a> on highway 101.
```

3. *The RDFa markup consists of URIs, which are used by applications for requesting additional information in RDF about the respective subject..*

```
dbp:Folks\%2C\_Washington
    rdfs:label      "Folks" ;
    geo:lat         "47.95"^xsd:double ;
    geo:long        "-124.38"^xsd:double ;
    foaf:depiction <http://upload.wikimedia.org/.../Forks_WA.jpg> ;
    rdfs:seeAlso   <http://maps.google.de/maps?ll=47.95,-124.38>  .
```

4. *Finally, this information is aggregated or visualized in mashups (see Figures 9.2a, 9.2b, 9.3, 9.6).*

9.3.2 Position

Inspired by creating LOD perspectives on Web Pages and applications such as Sterntaler and Epiphany, Adrian [2010] proposed a position about the potentials of enriching Web documents with information from LOD by generating RDFa markup. Three exemplary user stories illustrate the impact of RDF-based IE on three use cases.

Provide Sarah with additional product offers and reviews. Sarah is a PHD student. shes wants to buy a laptop and has a budget of 600$. Sarah has a tight schedule, so her plan is to search online to compare different offers regarding to product properties and existing reviews from customers. Fortunately, Sarah knows LOD and she is happy to see a LOD wrapper that gives her access to `Amazon.com` data in RDF format. She knows Amazon possess a large knowledge base about products and vendors, their offers, and user generated ratings and experience reports about using these products. Sarah likes Amazon and its marketplace, but wants to give all online vendors a chance. Finally, she wants the cheapest offer for the best laptop she can get with 600$. Therefore, she wants a projection of Amazon's product data to products mentioned in Web pages of online shops she found while searching Google products.

Stimulate Pete's imagination while reading a book Pete is a high school student. He is reading Stephanie Meyer's exciting vampire thriller on his iPad. Pete loves to have a more colorful imagination about concrete sets and locations where actions take place in. Therefore, he installed a fancy App that uses data from the DBpedia and the LinkedGeoData to produce a mashup on Google maps and Google Streetview. This app provides him with scenery pictures, satellite images, and links to additional background information about heritages, famous buildings, battlegrounds, etc. Pete enjoys looking at real pictures and maps around the primeval forests near the small town Folks in Washington, USA, which is the set of this book on vampires, American natives, and werewolves.

Keep Tom up-to-date with relevant posts from Twitter and Facebook. Tom is an emergent entrepreneur. He really knows about the power of social platforms like Twitter or Facebook. Tom knows many experts, friends, and customers inside these platforms and he likes to know about their opinion about some new technologies and products Tom is reading about in blogs or other Web pages. Tom installed a browser plugin that uses the RDF data published by Twitter and Facebook to receive the latest tweets, comments, and blog entries from twitter and Facebook from his contacts about topics mentioned in these Web pages.

9.3.3 Conclusion

Commercial services, such as Zemanta and Open Calais, show that the OBIE functionality, even when being tied to a single domain of concern, provides great potentials for indexing and enriching text content. Adrian et al. [2010] describe this as bridging the gap between the Web of Documents and the Web of Data. Compared to specialized OBIE, the presented RDF-based IE approach is fa more general as it can adapt to any kind of domain of concern as long as the contained data is published as RDF graph. More specifically, Adrian and Dengel [2011] describe the use case of utilizing RDF from LOD datasets about a certain field of knowledge (e.g., information about products, encyclopedias, music) as a perspective that provides a specialized point of view on Web documents by emphasizing those parts of the document content, for which additional information exists. Agents are enabled to requesting RDF graphs within the Web of Data for more information about recognized semantic entities.

Epiphany, *Sterntaler*, and the tools Sarah, Pete, and Tom are using, utilize RDF data that is published as LOD for enriching semantic entities with formal markup. This markup explicates the reference between a phrase in text and the instance within a data set. The tools of Sarah, Pete, and Tom depend on different LOD sources enriching the document content with additional information. The tools are based on the content enrichment process, which is made possible by the proposed RDF-based IE approach and LOD.

10 Concluding Information Extraction on the Semantic Web

It's Not Information Overload. It's Filter Failure.

(Clay Shirky, 2008)

This work is introduced by Clay Shirky's quote on information overload. He demands effective filtering mechanisms for avoiding effects of information overloading. RDF based IE filters natural language text content by means of domain knowledge from an existing RDF graph. The proposed approaches facilitate the implementation of efficient information filtering tools demanded by Shirky. Hereby, the underlying research combines the state-of-the-art from two disciplines. On the one hand, it utilizes Semantic Web technology formalizing information for an open exchange on the Web. On the other hand, it extends IE methods of Computational Linguistics. Finally, the presented approach extends the Semantic Web technology stack by providing RDF views on the natural language content of Web pages.

This chapter summarizes and concludes the presented RDF-based IE technologies. Section 10.1 recapitulates the initial research hypotheses and summarizes the efforts spent on proving these. Section 10.2 reviews the lessons learned on performing IE on the Semantic Web. The conclusion of the major outcomes and contributions is described in Section 10.4. Section 10.5 discusses the predicted impact of RDF-based IE on the Semantic Web handling and filtering information in near future search and browsing applications.

10.1 Summary

This work investigated IE approaches utilizing information, which is formalized in terms of RDF. The underlying motivation and intention of the activities spent on this area of research was to gain insights on how to incorporate RDF into IE-systems in order to verify the following hypotheses (see Section 1.2):

The utilization of RDF into IE ...

H.1 ... enhances the IE process.

H.2 ... facilitates design of IE-interfaces.

H.3 ... involves domain adaptability.

These hypotheses result from a state-of-the-art analysis on open challenges in IE and Semantic Web research (see Chapters 2 and 3). In common, the existing related work focuses on single domains of concerns and does not provide any research insights on how each single component of an RDF graph can be used for supporting IE algorithms (see Chapter 4). Computational foundations were investigated and evaluated for passing information from RDF graphs to IE algorithms by using tools, such as matrices from linear algebra, metrics from statistics, probabilities from stochastics,and their combined use in form of machine learning models (see Chapter 5). These foundations facilitate clearly defining the Semantic Entity Recognition problem. Pre-processing techniques were proposed for representing and rehashing information from RDF graphs for being utilized by information extractors (see Chapter 6). Covering the complete IE process, particular information extractors were enhanced by utilizing the information from RDF graphs. They were evaluated on different RDF graphs and document corpora (see Chapter 7). With respect to the input-output-interface of respective IE systems, IE templates in terms of SPARQL queries as well as RDF serializations of IE results were developed (see Chapter 8). The overall RDF-based IE approach resulted in a series of applications, namely: *SCOOBIE*, *Epiphany*, and *Sterntaler* (see Section 9.1).

10.2 Discussion

The efforts spent on investigating the use of RDF within IE approaches involved issues, which can be summarized by the following question:

"Does an RDF graph, formally describing a domain of concern, provide enough information for information extractors to solve their respective goals?"

The answer is, it depends on the nature of the respective RDF graph. In Section 9.1.2, some natures of the DBpedia graph were illustrated, which irritated the resolution and disambiguation of instances. The experiments performed on the datasets, i.e., *DBpedia*, *BBC music*, and *BBC nature*, revealed that the quality of information extractors is influenced by the specific graph content. Of course, this is caused by the different statistics and model, which result from pre-processing these RDF graphs. The thresholds used for clustering correlating classes or computing with probabilities in Markov chains differ between RDF graphs. In addition, the metrics used for disambiguating or rating recognized instances may also perform differently on different RDF graphs. Developers should be aware of the fact that, even though RDF-based IE increases the amount of adaptability, the exchange of RDF graphs always involves a recalibration of the IE system.

An interesting result of the relevance rating experiment (see Section 7.9.9) was the increase of result quality when applying a combined usage of text corpus and graph based metrics. It raises the question, if a combined usage of the statistics from text corpora and an RDF graphs increases the performance of information extractors, in general.

10.3 Lessons-learned

In this work, many lessons about the natures of utilized RDF graphs could be learned when applying these within information extractors:

- Nearly all RDF graphs in the LOD cloud apply `rdfs:label` as basic naming property for all kind of instances and classes. However, modeling different naming properties to different classes facilitates the classification of named entities. There is much more semantic in recognizing in text a person name than just a name. Anyway, persons do have different names than companies, places, products, or songs. Hence, classes on a taxonomic level such as persons, places, or organizations should possess their own naming properties which could be subsumed by `rdfs:label`.
- The classification of instances should always be grounded on shared datatype and object properties. The reason is that the distinction between politicians and artists without using specialized datatype properties or object properties or respective values cannot be recognized by any machine learning model. This was the reason for clustering correlating classes in the pre-processing step (see Section 6.3).
- The nature of any formal instance is defined by its possessed properties. Adding specialized relationships with restrictions on classes of domain and range between instances increases the degree of formal semantics describing the nature of respective instances. This facilitates the application of link-based graph algorithms for classifying, disambiguating, and rating recognized instances.

- An `owl:sameAs` unification should be performed before utilizing the RDF graph in IE applications. This object property indicates that two URI references actually refer to the same thing: the individuals have the same "identity" [Bock et al., 2009]. Hence, a unification of both individuals results in a merged number of properties and values. It also avoids a disambiguation, as both URI represent the same instance.

- Classes, datatype and object properties without any instantiations do not carry any meaning that could be used for enhancing information extractors. The counted bias of instantiated vocabulary items (properties or classes) in RDF graphs is important as it influences resulting statistics. Uniform distributions of classes and properties propose a best possible utilization within information extractors. This holds especially when predicting new classes or properties.

10.4 Conclusion

In general, the activities performed in this work (Chapters 6 to 8) result in a number of scientific contributions to the Semantic Web and Information Extraction research. These are summarized in the following list. For a more detailed description, please refer to Section 1.3.

C.1 Incorporating RDF into IE enables information extractors to utilize additional background knowledge.

C.2 Incorporating RDF into IE involves the extension of traditional NER.

C.3 By utilizing terms from the formal vocabularies, which are used in RDF data sets, SPARQL queries are applied for specifying IE templates that describe which types of information the IE-system should extract from text.

C.4 In order to deploy the application of RDF-enhanced Information Extraction (IE) into the general architecture of a Semantic Web, this work illustrates how to formalize extraction results in RDF.

C.5 A method was developed to use the knowledge of an RDF graph for automatically labeling a text corpus with named entity classifications.

C.6 The investigated methods and applications are adaptable to various RDF graphs, as they were evaluated successfully on a number of different RDF graphs.

These conclusions relate to the originating research hypotheses. On a more technical level, the evaluation results provided us with the following realizations:

- The matrix representation of RDF graphs is suitable to perform efficient computations on contained semantics (see Section 5.4.2).
- The application of prefix hashed suffix arrays to recognizing property values in text scales to dimensions up to hundreds of millions of literals in RDF graphs (see Section 7.4).
- The representation of RDF graphs in the proposed relational database schema ensures a scaling, caching, querying access to information from RDF graphs within the scope of information extractors (see Section 6.2).
- The proposed clustering of correlating classes based on hierarchical clustering on a correlation matrix produces excellent results (see Section 6.3).
- The use of link-based metrics in combination with document-based statistics allows the rating of recognized instances by relevance. Such a rating is suitable for filtering irrelevant instances or for selecting the top rated k results (see Section 7.6).

Finally, the results of this research extended the state-of-the-art of the Semantic Web technology stack by providing RDF views on the natural language content of Web pages. The developed SCOOBIE system is published as Open Source at `https://github.com/benjamin-adrian/scoobie`.

10.5 Future Work

The research on Semantic Web based IE approaches is still emerging. The evaluation of existing services in Section 9.1.2 reveals that currently, the performance of such systems is not ready for being used in enterprise applications. Because of experiences of this work, we recommend investigating on the following research objectives.

10.5.1 Short-term investigations

Tensor representation of RDF graphs Instead of using a two-dimensional matrix for representing RDF graphs (see Section 5.4.2), Franz et al. [2009] proposed using three-dimensional tensors. Their research objective was focusing on rating RDF triples in RDF graphs by relevance. Therefore, they generalized the Eigenvalue-based PageRank algorithm to work on three dimensional tensors. It has to be investigated if the representation in tensors enhances the disambiguation and rating, as well as the prediction of new relations between recognized instances.

Optimal count of clustering correlating classes Section 6.3 described the clustering of correlating classes. The resulting approaches have to be parametrized specifying

the number of clusters. Sugar and James [2003] describe an approach to automatically estimating a good number of clusters. It is based on k-means clustering, but we recommend transferring it to the hierarchical clustering, which was applied in this work.

Combining graph-based disambiguation with text-based statistics Mendes et al. [2011], the researchers of the DBpedia Spotlight system implemented the disambiguation of instances on the basis of textual descriptions about each instance in combination with the textual context around each recognized entity in text. It seems promising combining the graph-based disambiguation (see Section 7.5) evaluated in this work with the text-based approach of DBpedia Spotlight.

Train sequence taggers for classifying named entities The proposed training of named entity classifiers, as described in Section 7.7, is based on a relational maximum entropy classifier (see Section 5.4.7). It seems promising evaluating the use of sequence taggers, such as a CRF (see Section 5.4.8), which is an extension of a maximum entropy model, for classifying named entities.

Application of semantic roles Computational Linguistics comprise the notion of formalizing the meaning of events in sentences by introducing constraints on the arguments of event predicates [Jurafsky and Martin, 2008]. Semantic roles define signatures on predicates specifying, for example, "is employed at" to associate an agent of kind person with an agent of kind organization. Palmer et al. [2005] describe Proposition Bank, an annotated corpus of semantic roles. The association of roles from Proposition Bank with object properties from an RDF graph should facilitate the classification of named entities in text and the recognition of relations between named entities in a sentence.

Semi-supervised learning Hearst [1992] proposed a semi-supervised learning approach for recognizing taxonomic relationships. The automatically labeling of training data described in Section 6.8 may be extended with similar bootstrapping techniques in order to enhance coverage of labeled entities in corpus.

Distributions of standard test corpora The algorithms developed and investigated in this research are evaluated on different corpora (see Section 7.9.1). All efforts spent on creating these corpora were part of this research. The OBIE research community would benefit from providing a series of gold standard corpora. A significant system comparison can only be provided by evaluating OBIE systems and algorithms based on such gold standard corpora. Hence, we strongly recommend its creation.

Deploy SPARQL endpoints on the basis of SCOOBIE Following the contribution of this work, we recommend deploying SCOOBIE as open Web service. It should

provide one or multiple SPARQL endpoints relating to RDF graphs, such as DBpedia or Freebase.

10.5.2 Long-term investigations

In addition to these technical issues, investigating the following rather general issues, would facilitate the implementation of RDF based IE systems:

Extend ontology languages to represent proper names. In Section 5.1, the sign theory of Peirce is combined with the semantics of proper names from text and instances from RDF graphs. Unfortunately, it is not possible to distinguish the natures of datatype properties in clearly representing either labels (e.g., "sweetheart"), attributes (e.g., "1.73m"), or proper names (e.g., "Benjamin Adrian") in RDF graphs. Very frequently `rdfs:label` or derived datatype properties are used for describing labels and names for all classes of instances within the RDF graph. However, assigning well-defined proper naming datatype properties for each class preserves a higher degree of formalism. Based on a match between a noun phrase in text and such a proper naming datatype property value, the information extractor is capable to infer the type of the respective semantic entity, directly. This facilitates the disambiguation of word senses and would enhance the performance of the automatically labeling of corpus data.

Extend the functional capabilities of SPARQL. Extending the Semantic Web with a query language, the SPARQL specification is defined upon the stateless HTTP protocol. This results in the absence of cursor concepts, which leads to a large consumption of a channel's bandwidth and client's memory when requesting large portions of data. This is the case, when looking for matching datatype property values in text. Here, hundreds of thousands of results may be received at once, when matching the word "gold" with the, for example, the DBpedia. Such a massive amount of data clearly exceeds the capacity and answer time of networks and applications. Hence, we used a relational database as back-end. However, as soon as SPARQL endpoints provide functionality for such scalable requests, they could serve as native back-ends for RDF based IE systems. This would increase the adaptability of such systems, as they would be able connecting to any SPARQL endpoint, which is available on the Web.

Complete the DOM API to create elements within plain text content. The serialization of IE results in RDFa requires the creation of HTML elements around phrases in text. The Document Object Model (DOM) of HTML is an API to let programs access, create, and modify elements. Unfortunately, it lacks the support for creating elements, such as `<DIV>` or `<SPAN>`, in plain text. These elements are required to append RDF in attributes. Hence, adding such a functionality to the World

Wide Web Consortium (W3C)'s DOM specification facilitates the serialization of IE results in Web pages via RDFa. Of course, such a modification creates development costs for browser vendors and other API implementers. Therefore, it is considered as a long-term goal.

Investigate the recognition of structural natures of RDF graphs. Using information from RDF graphs, i.e., proper names of instances and links between instances, can be facilitated if some natures of the RDF graph are explicitly known. The following questions should give an impression on the impact of such an investigation:

Is the graph between instances fully connected or does it contain a number of unconnected clusters? The RDF graph of DBpedia is fully connected, the graphs of BBC nature and BBC music not. As shown in Sections 7.9.6 and 7.9.9, connectivity influences the effective application of link metrics to disambiguating and rating recognized instances in text.

How many class hierarchies are used within the RDF graph? How balanced, in numbers of instantiations, are these hierarchies? Do they overlap? The clustering of and labeling with classes in Chapter 6 illustrates the effects of class hierarchies to IE tasks.

10.6 Acknowledgments

Activities on this work accompanied a series of research projects, which resulted in a list of contributions.

DocuTag, German research funding, Stiftung Rheinland-Pfalz für Innovation, [Adrian et al., 2008a]

iDocument, German research funding, Stiftung Rheinland-Pfalz für Innovation, [Adrian et al., 2009b]

Nepomuk, EU FP6 research funding, Grant FP6-027705, [Dengel and Adrian, 2011]

Perspecting, German research funding, BMBF, Grant 01IW08002, [Adrian et al., 2010], [Dengel and Adrian, 2011]

REMIX, German research funding, Zentrales Innovationsprogramm Mittelstand (ZIM), Grant KF2013005SM9, [Adrian and Dengel, 2011]

SemoPad, German research funding, Stiftung Rheinland-Pfalz für Innovation, [Adrian and Schwarz, 2011]

Bibliography

B. Adida, I. Herman, M. Sporny, and M. Birbeck. RDFa 1.1 Primer, rich structured data markup for web documents. Technical report, World Wide Web Consortium, 4 2011. URL http://www.w3.org/TR/rdfa-primer/. 22, 195

B. Adrian. Potentials of enriching the Web of Documents with Linked Data by generating RDFa markup. In V. Presutti, F. Scharffe, and V. Svatek, editors, *Proceedings of the 1st Workshop on Knowledge Injection into and Extraction from Linked Data. Workshop on Knowledge Injection into and Extraction from Linked Data (KIELD-2010)*, volume 631, pages 66–67. CEUR-WS, 2010. 195, 214

B. Adrian and A. Dengel. Believing Finite-State cascades in Knowledge-based Information Extraction. In A. Dengel, K. Berns, T. Breuel, F. Bomarius, and T. Roth-Berghofer, editors, *KI 2008: Advances in Artificial Intelligence. German Conference on Artificial Intelligence (KI), Kaiserslautern, Germany*, volume 5243 of *Lecture Notes in Computer Science, LNCS*, pages 152–159. Springer, 2008. ISBN 978-3-540-85844-7. 38, 39

B. Adrian and A. Dengel. Linked Open Data Perspectives: Incorporating Linked Open Data into Information Extraction on the Web. *it - Information Technology*, 53(3): 117–124, 5 2011. 206, 215, 224

B. Adrian and S. Schwarz. Using Suffix Arrays for Efficient Recognition of Named Entities in Large Scale. In A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, editors, *Knowlege-Based and Intelligent Information and Engineering Systems*, volume 6882 of *Lecture Notes in Computer Science*, pages 420–429. Springer, 2011. ISBN 978-3-642-23862-8. 137, 138, 224

B. Adrian, H. Maus, and A. Dengel. DocuTag - Semantische Dienste für das Tagging von Dokumenten in Unternehmen: Kompetenzzentrum für das Büro der Zukunft. In B. Klempt, D. J. Ohl, C. Janssen-Neumann, U. Mayer, and G. Fischer, editors, *Stiftung Rheinland-Pfalz für Innovation Jahresbericht 2007*, pages 39–41. Stiftung Rheinland-Pfalz für Innovation, 2008a. 192, 224

B. Adrian, G. Neumann, A. Troussov, and B. Popov, editors. *Proceedings 1st International and KI-08 Workshop on Ontology-based Information Extraction Systems. Ontology-based Information Extraction Systems (OBIES-08), located at KI 2008, September 23-26, Kaiserslautern, Germany*, volume 400, 2008b. DFKI, CEUR. 66

B. Adrian, B. Forcher, T. Roth-Berghofer, and A. Dengel. Explaining Ontology-Based Information Extraction in the NEPOMUK Semantic Desktop. In T. Roth-Berghofer, N. Tintarev, and D. B. Leake, editors, *Workshop 10@IJCAI-09: Explanation-aware Computing (ExaCt 2009). International Joint Conference on Artificial Intelligence (IJCAI-09), July 11-12, Pasadena,, California, United States*, pages 94–101. AAAI, 7 2009a. 212

B. Adrian, J. Hees, L. van Elst, and A. Dengel. iDocument: Using Ontologies for Extracting and Annotating Information from Unstructured Text. In B. Mersching, M. Hund, and Z. Aziz, editors, *KI 2009: Advances in Artificial Intelligence. German Conference on Artificial Intelligence (KI-2009), September 15-18, Paderborn, Germany*, volume 5803 of *Lecture Notes in Artificial Intelligence, LNAI*, pages 249–256. Springer-Verlag, Heidelberg, 9 2009b. ISBN 978-3-642-04616-2. 224

B. Adrian, M. Klinkigt, H. Maus, and A. Dengel. Using iDocument for Document Categorization in Nepomuk Social Semantic Desktop. In A. Paschke, H. Weigand, W. Behrendt, K. Tochtermann, and T. Pellegrini, editors, *Proceedings of I-KNOW 09 and I-SEMANTICS 09. International Conference on Semantic Systems (iSemantics-09), September 2-4, Graz, Austria*, pages 638–643. Verlag der Technischen Universität Graz, Graz, 9 2009c. ISBN 978-3-85125-060-2. 211

B. Adrian, H. Maus, and A. Dengel. iDocument: Using Ontologies for Extracting Information from Text. WM2009: 5th Conference on Professional Knowledge Management, 3 2009d. 196

B. Adrian, H. Maus, M. Kiesel, and A. Dengel. Towards Ontology-based Information Extraction and Annotation of Paper Documents for Personalized Knowledge Acquisition. In K. Hinkelmann and H. Wache, editors, *WM2009: 5th Conference on Professional Knowledge Management. Conference on Professional Knowledge Management (WM-2009), March 25-27, Solothurn, Switzerland*, volume P-145 of *Lecture Notes in Informatics, LNI*. GI, Gesellschaft für Informatik, Bonn, 3 2009e. ISBN 978-3-88579-239-0. 209

B. Adrian, , J. Hees, I. Herman, M. Sintek, and A. Dengel. Epiphany: Adaptable RDFa Generation Linking the Web of Documents to the Web of Data. In *Proceedings of EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses*, Lecture Notes in Informatics, LNI. Springer-Verlag, Heidelberg, October 2010. 147, 206, 215, 224

N. I. Al-Rajebah and H. S. Al-Khalifa. Semantic Relationship Extraction and Ontology Building using Wikipedia: A Comprehensive Survey. *International Journal of Computer Applications*, 12(3):6–12, December 2010. Published By Foundation of Computer Science. 151

C. Allauzen and M. Mohri. N-Way Composition of Weighted Finite-State Transducers. *Int. J. Found. Comput. Sci.*, 20(4):613–627, 2009. 39

J. Allen. *Natural Language Understanding*. Benjamin/Cummings Publishing Company, Inc., redwood City, CA 94065, 2 edition, 1994. ISBN 978-0805303346. 30

J. Alpert and N. Hajaj. We knew web was big. Official Google blog, July 2008. URL http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html. 21

D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. FASTUS: A Finite-State Processor for Information Extraction From Real-World Text. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 1993. 38

D. E. Appelt and D. J. Israel. Introduction to Information Extraction Technology. A tutorial prepared for IJCAI-99, Stockholm, Schweden, 1999. URL http://www.ai.sri.com/~appelt/ie-tutorial/IJCAI99.pdf. 38

N. Bach and S. Badaskar. A review of relation extraction, 2007. URL http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf. 47, 48

D. Becket. RDF/XML): Syntax Specification (Revised). Technical report, World Wide Web Consortium, 2 2004. URL http://www.w3.org/TR/rdf-syntax-grammar/. 53

D. Beckett and T. Berners-Lee. Turtle - Terse RDF Triple Language. Team submission, W3C, 2007. URL http://www.w3.org/TeamSubmission/turtle/. http://www.w3.org/TeamSubmission/turtle/. 26, 53

T. Berners-Lee. Linked data. World wide web design issues, July 2010. URL http://www.w3.org/DesignIssues/LinkedData.html. 61

T. Berners-Lee and D. Connolly. Notation3 (N3): A readable RDF syntax. Technical report, World Wide Web Consortium, 1 2008. URL http://www.w3.org/TeamSubmission/n3/. 53

T. Berners-Lee and M. Fischetti. *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco, sep 1999. ISBN 0062515861. 25

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001. ISSN 0036-8733. 22, 51, 52

C. Bizer and R. Cyganiak. The TriG Syntax. Technical report, FU Berlin, 7 2007. URL http://www.wiwiss.fu-berlin.de/suhl/bizer/TriG/Spec/. 58, 194

C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009a. 61

C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics*, 7:154–165, September 2009b. ISSN 1570-8268. 58, 119

C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and M. Smith. OWL 2 web ontology language structural specification and functional-style syntax. Technical report, World Wide Web Consortium, 10 2009. URL http://www.w3.org/TR/owl2-syntax/. 57, 79, 220

K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to meet new challenges in language engineering. *Natural Language Engineering*, 10(3–4):349–373, 2004. ISSN 1351-3249. 38, 66, 69

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, 2002. 41, 136

T. Breuel. The OCRopus Open Source OCR System. In B. Yanikoglou and K. Berkner, editors, *Proceedingsof the Document and Retrival XV, IS&T/SPIE 20th Annual Symposium 2008. DRR-2008, January 16-31, San Jose, CA, United States*, volume 6815. SPIE, 2008. 209

C. Brewster, F. Ciravegna, and Y. Wilks. Background and Foreground Knowledge in Dynamic Ontology Construction Viewing Text as Knowledge Maintenance. In R. Dieng-Kuntz and F. Gandon, editors, *KCAP 2003 Workshop on Knowledge Management and the Semantic Web*, pages 9–16. KCAP, 2003. A revised version of a paper presented at SIGIR 03. 65

D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF schema. Technical report, World Wide Web Consortium, 2 2004. URL http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. 58, 78

D. Brickley and L. Miller. FOAF Vocabulary Specification 0.98, January 2010. URL http://xmlns.com/foaf/spec/20100809.html. 58

J. Broekstra, A. Kampman, and F. V. Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. Hendler, editors, *The Semantic Web – ISWC 2002*, Lecture Notes in Computer Science, pages 54–68. Springer Berlin / Heidelberg, 2002. 100

P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, July 2005. 65, 67, 72, 151

P. Buitelaar, P. Cimiano, S. Racioppa, and M. Siegel. Ontology-based Information Extraction with SOBA. In *Proceeding of LREC*, Genoa, Italy, 5 2006. 69

J. J. Carroll and P. Stickler. RDF Triples in XML. In *Proceedings of the Extreme Markup Languages 2004 Conference*, 2004. URL `http://conferences.idealliance.org/extreme/html/2004/Stickler01/EML2004Stickler01.html`. 53, 58

J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 613–622, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. 58

N. Chomsky. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113– –124, September 1956. ISSN 0096-1000. 31, 39

P. Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387306323. 15, 66, 67, 72

P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning Taxonomic Relations from Heterogeneous Sources. In *Proceedings of the ECAI Ontology Learning and Population Workshop*, 2004. 46, 48

R. Cole, editor. *Survey of the state of the art in human language technology*. Cambridge University Press, New York, NY, USA, 1997. ISBN 0-521-59277-1. 40

J. Cowie and W. Lehnert. Information Extraction. *Communications of the ACM*, 39(1): 80–91, 1996. 35, 49

J. Cowie and Y. Wilks. Information Extraction. In R. Dale, H. Moisl, and H. Somers, editors, *Handbook of Natural Language Processing*. Marcel Dekker, New York, 2000. 49

H. Cunningham. Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, 5:665–677, November 2006. 49

H. Cunningham, D. Maynard, and V. Tablan. JAPE: A Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, 2000. 43

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *40th Anniversary Meeting of the ACL, Proceedings*, 2002. 43

G. DeJong. Prediction and substantiation: A new approach to natural language processing. *Cognitive Science*, 3(3):251–273, 1979. ISSN 0364-0213. 35

G. F. DeJong. An overview of the FRUMP system. In W. G. Lehnert and M. H. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum, Hillsdale, NJ, 1982. 35

K. Dellschaft and S. Staab. Strategies for the Evaluation of Ontology Learning. In *Proceeding of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 253–272, Amsterdam, The Netherlands, 2008. IOS Press. ISBN 978-1-58603-818-2. 121

A. Dengel and B. Adrian. Helping People Remember: Coactive Assistance for Personal Information Management on a Semantic Desktop. In A. Fred, J. L. G. Dietz, K. Liu, and J. Filipe, editors, *Knowledge Discovery, Knowlege Engineering and Knowledge Management*, volume 128 of *Communications in Computer and Information Science*, pages 3–16. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-19032-2. 210, 224

B. DuCharme. *Learning SPARQL*. Oreilly & Assoc. Inc., 2011. ISBN 1449306594. 189, 197

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, second edition, 2001. 26, 87, 88, 90, 94, 97

T. Dunning. Statistical identification of language. Technical report, MCCS 94-273, New Mexico State University, 1994. 40

S. Ebert, M. Liwicki, and A. Dengel. Ontology-based information extraction from handwritten documents. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition. International Conference on Frontiers in Handwriting Recognition (ICFHR-10), November 16-18, Kolkata, India*, pages 483–488, 2010. 210

P. Elango. Coreference Resolution: A Survey. Technical report, University of Wisconsin Madison, 2005. 46

D. W. Embley, D. M. Campbell, R. D. Smith, and S. W. Liddle. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *CIKM '98: Proc. of the 7th international conference on Information and knowledge management*, pages 52–59, New York, NY, USA, 1998. ACM. ISBN 1-58113-061-9. 66, 69

D. W. Embley, C. Tao, and S. W. Liddle. Automatically Extracting Ontologically Specified Data from HTML Tables of Unknown Structure. In *Proceedings of the 21st International Conference on Conceptual Modeling*, ER '02, pages 322–337, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44277-4. 70, 71

B. Endres-Niggemeyer. Ontology-based information extraction in agents' hands. In *Proceedings 1st International and KI-08 Workshop on Ontology-based Information Extraction Systems*, volume 400, pages 15–21. DFKI, 2008. 70

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. 43

B. Forcher, B. Adrian, and T. Roth-Berghofer. Explanations in the information extraction system iDocument. *Künstliche Intelligenz (KI)*, Schwerpunkt: Erklärungen (2/08):32–34, 5 2008a. 212

B. Forcher, B. Adrian, and T. Roth-Berghofer. Explanation Styles in iDocument. In T. Roth-Berghofer, S. Schulz, D. Bahls, and D. B. Leake, editors, *Proceedings of the Third International Workshop on Explanation-aware Computing. International Workshop on Explanation-aware Computing, in Conjunction with 3rd, located at 18th European Conference on Artificial Intelligence ECAI 2008, July 21-22, Patras, Greece*, volume 39, pages 144–156. CEUR-WS.org, Aachen, 2008b. 211

T. Franz, A. Schultz, S. Sizov, and S. Staab. TripleRank: Ranking Semantic Web Data by Tensor Decomposition. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 213–228, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04929-3. 221

D. Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, Nov. 1998. URL `http://www.cs.cmu.edu/afs/cs/user/dayne/www/ps/diss-freitag.ps`. 43

D. Freitag. Machine Learning for Information Extraction in Informal Domains. *Mach. Learn.*, 39:169–202, May 2000. ISSN 0885-6125. 43

R. Gaizauskas and Y. Wilks. Information Extraction: Beyond Document Retrieval. *Journal of Documentation*, 54(1):70–105, 1998. 49

B. Greene and G. Rubin. Automatic grammatical tagging of english. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, 1981. 41

R. Grishman. Information Extraction: Techniques and Challenges. In *SCIE '97: Int. Summer School on IE*, pages 10–27, London, UK, 1997. Springer. ISBN 3-540-63438-X. 49

R. Grishman. The Proteus Project at New York University, 2002. URL `http://nlp.cs.nyu.edu`. 33, 34

R. Grishman and B. Sundheim. Message Understanding Conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics*, pages 466–471, Morristown, NJ, USA, 1996. Association for Computational Linguistics. 35, 38

T. Groza, S. Handschuh, K. Moeller, G. A. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjónsdóttir. The NEPOMUK Project - On the way to the Social Semantic Desktop. In *Proceedings of International Conferences on new Media technology (I-MEDIA-2007) and Semntic Systems (I-SEMANTICS-07), Graz, Austria, September 5-7.*, pages 201–210, 9 2007. 211

T. R. Gruber. A Translation Approach to Portable Ontologies Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993. 59

S. Handschuh, S. Staab, and F. Ciravagna. S-CREAM — Semi-automatic CREAtion of Metadata. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Proceedings*, pages 358–372, Siguenza, Spain, 2002. 69

P. Hayes. RDF Semantics. Technical report, World Wide Web Consortium, 2 2004. URL `http://www.w3.org/TR/rdf-mt/`. 55

M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. 45, 48, 222

P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009. 26

J. Hobbs and D. Israel. Principles of template design. In *HLT '94: Proc. of the workshop on Human Language Technology*, pages 177–181, Morristown, NJ, USA, 1994. ACL. ISBN 1-55860-357-3. 37, 49

J. R. Hobbs. The generic information extraction system. In *MUC5 '93: Proceedings of the 5th conference on Message understanding*, pages 87–91, Morristown, NJ, USA, 1993. ACL. ISBN 1-55860-336-0. 37

D. Huynh, S. Mazzocchi, and D. Karger. Piggy Bank: Experience the Semantic Web inside your web browser. *Web Semantics*, 5(1):16–27, 2007. ISSN 1570-8268. 70

N. Ireson, F. Ciravegna, M. E. Califf, D. Freitag, N. Kushmerick, and A. Lavelli. Evaluating machine learning for information extraction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 345–352, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. 43, 176

I. Jacobs and N. Walsh. Architecture of the World Wide Web, Volume One. Recommendation, W3C, 2004. URL http://www.w3.org/TR/webarch/. 22, 25

P. Jain, P. Hitzler, P. Z. Yeh, K. Verma, and A. P. Sheth. Linked Data is Merely More Data. *AAAI Spring Symposium "Linked Data Meets Artificial Intelligence"*, 2010. 62

E. T. Jaynes. Information Theory and Statistical Mechanics. *The Physical Review*, 106 (4):620–630, Mai 1957. 93

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008. ISBN 0130950696. 25, 26, 30, 33, 38, 40, 42, 47, 48, 93, 222

J. Kärkkäinen, P. Sanders, and S. Burkhardt. Linear work suffix array construction. *J. ACM*, 53:918–936, November 2006. ISSN 0004-5411. 85

J. Kleb and A. Abecker. Entity Reference Resolution via Spreading Activation on RDF-Graphs. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, Proceedings, Part I*, volume 6088 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2010. ISBN 978-3-642-13485-2. 144

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46: 604–632, September 1999. ISSN 0004-5411. 92

G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, World Wide Web Consortium, 2 2004. URL http://www.w3.org/TR/rdf-concepts/. 55, 57

Y. Koren and R. M. Bell. Advances in Collaborative Filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer, 2011. ISBN 978-0-387-85819-7. 151

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 2001. 95

N. Kushmerick. Finite-State Approaches to Web Information Extraction. In *SCIE*, pages 77–91, 2002. 38

M. Labsky. *Information Extraction from Websites using Extraction Ontologies*. PhD thesis, University of Economics Prague, 2008. 71

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. 95

A. N. Langville, Carl, and D. Meyer. A survey of eigenvector methods of web information retrieval. *SIAM Review*, 47(1):135–161, March 2009. 92

Y. Li and K. Bontcheva. Hierarchical, perceptron-like learning for ontology-based information extraction. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 777–786, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. 68, 69

A. Maedche and S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16:72–79, March 2001. ISSN 1541-1672. 66

A. D. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, Norwell, MA, USA, 2002. ISBN 0792376560. 72

C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715. 114, 146, 158

F. Manola, E. Miller, and B. McBride. RDF Primer. Technical report, World Wide Web Consortium, 2 2004. URL `http://www.w3.org/TR/rdf-primer/`. 22, 53, 54, 57

A. K. McCallum. MALLET: A Machine Learning for Language Toolkit., 2002. URL `http://mallet.cs.umass.edu`. 38, 136, 203

P. Mendes, M. Jakob, A. Garcia-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics)*, September 2011. 70, 137, 197, 222

D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, January 2007. ISSN 0378-4169. 38, 43, 45

R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41:10:1–10:69, February 2009. ISSN 0360-0300. 46

C. Nedellec and A. Nazarenko. Ontologies and Information Extraction. *CoRR*, abs/cs/0609137, 2006. 72

A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, pages 903–910, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-812-5, 978-1-558-60812-2. 92

K. Nigam, J. Lafferty, and A. Mccallum. Using Maximum Entropy for Text Classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999a. 93, 94

K. Nigam, J. Lafferty, and A. Mccallum. Using Maximum Entropy for Text Classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999b. 41

OpenLink Software Documentation Team. *16. RDF Data Access and Data Management*. OpenLink Software, August 2011. URL http://docs.openlinksw.com/virtuoso/rdfdatarepresentation.html. last visited: 2001, 18 Aug. 100

E. Oren, S. Gerke, and S. Decker. Simple Algorithms for Predicate Suggestions Using Similarity and Co-occurrence. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, ESWC '07, pages 160–174, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-72666-1. 151

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120. 92

M. Palmer, D. Gildea, and P. Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31:71–106, March 2005. ISSN 0891-2017. 222

C. S. Peirce. *Mathematical Philosophy*, volume IV of *The New Elements of Mathematics*. Carolyn Eisele, ed., Mouton Publishers, The Hague, Netherlands, 1976. Humanities Press, Atlantic Highlands, 1976. ISBN 90 279 3035 X. 76, 116

A.-V. Pietarinen. Peirce's Pragmatic Theory of Proper Names. *Transactions of the Charles S. Peirce Society*, 46(3):341–363, 2010. 116

B. Popov, A. Kiryakov, D. Manov, D. Ognyanoff, and M. Goranov. KIM - Semantic Annotation Platform. In *Second International Semantic Web Conference (ISWC2003), Proceedings*, volume 124, pages 834–849. Springer-Verlag Heidelberg, 2003. ISBN 3-540-20362-1. 69

B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, and A. Kirilov. KIM – a semantic platform for information extraction and retrieval. *Natural Language Engineering*, 10 (3-4):375–392, 2004. 65, 68

E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C recommendation, World Wide Web Consortium, 2008. URL `http://www.w3.org/TR/rdf-sparql-query/`. 60

C. Reuschling, S. Agne, and A. Dengel. DynaQ - Faceted Search for Document Retrieval. In *DAS '10: Proceedings of the 9th IAPR International Workshop on Document Analysis Systems. IAPR International Workshop on Document Analysis Systems (DAS-10), June 9-11, Boston,, MA, United States*. ACM, 2010. 205

T. Roth-Berghofer and B. Adrian. From Provenance-awareness to Explanation-awareness—When Linked Data Is Used for Case Acquisition from Texts. In C. Marling, editor, *ICCBR 2010 Workshop Proceedings. Provenance-Aware Case-Based Reasoning: Applications to Reasoning, Metareasoning, Maintenance and Explanation (PACBR-2010), located at ICCBR 2010, July 19-22, Alessandria, Italy*, pages 103–106. Dipartimento di Informatica Università del Piemonte Orientale "A. Avogadro", Alessandria, 6 2010. 212

T. Roth-Berghofer, B. Adrian, and A. Dengel. Case Acquisition from Text: Ontology-based Information Extraction with SCOOBIE for myCBR. In I. Bichindaritz and S. Montani, editors, *Case-Based Reasoning Research and Development: 18th International Conference on Case-Based Reasoning. International Conference on Case-Based Reasoning (ICCBR-2010), in Conjunction with 18th, July 19-22, Alessandria, Italy*, volume 6176 of *Lecture Notes in Artificial Intelligence, LNAI*, pages 451–464. Springer Verlag, Heidelberg, 7 2010. ISBN 978-3-642-14273-4. 212

N. Sager. Syntactic analysis of natural language. *Advances in Computers*, 8:153–188, 1967. 35

N. Sager. *Natural Language Information Processing: A Computer Grammar of English and Its Applications*. Addison-Wesley Pub. Co., 1981. ISBN 0201067692. 35

H. Saggion, A. Funk, D. Maynard, and K. Bontcheva. Ontology-based Information Extraction for Business Applications. In *Proceedings of the 6th International Semantic Web Conference (ISWC 2007)*, Busan, Korea, November 2007. 69

L. Sauermann, G. A. Grimnes, M. Kiesel, C. Fluit, H. Maus, D. Heim, D. Nadeem, B. Adrian, and A. Dengel. Semantic Desktop 2.0: The Gnowsis Experience. In *International Semantic Web Conference. Volume 4273 of Lecture Notes in Computer Science*, number 4273 in Lecture Notes in Computer Science, pages 887–900. Springer, 11 2006. 210

R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ, 1977. 35

J. R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(03):417–424, 1980. 59

S. C. Shapiro. *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1992. ISBN 0471503053. 30

C. Shirky. It's Not Information Overload. It's Filter Failure. Keynote at Web 2.0 Expo, September 2008. URL http://www.web2expo.com/webexny2008/public/schedule/detail/4817. 21

M. Sintek, M. Junker, L. van Elst, and A. Abecker. Using Information Extraction Rules for Extending Domain Ontologies. In *Workshop on Ontology Learning*. CEUR-WS.org, 2001. 69

SPARQL Working Group. Sparql query language for rdf, 2008. URL http://www.w3.org/2001/sw/wiki/index.php?title=SPARQL&oldid=1374. 60, 196

M. M. Stark and R. F. Riesenfeld. Wordnet: An electronic lexical database. In *Proceedings of 11th Eurographics Workshop on Rendering*. MIT Press, 1998. 37

P. Stickler. CBD - concise bounded description. Technical report, World Wide Web Consortium, June 2005. URL http://www.w3.org/Submission/CBD/. 55

C. A. Sugar and G. M. James. Finding the Number of Clusters in a Dataset: An Information-Theoretic Approach. *Journal of the American Statistical Association*, 98 (463):750–763, 2003. ISSN 01621459. 105, 222

C. Sutton and A. McCallum. An Introduction to Conditional Random Fields for Relational Learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006. URL http://www.cs.umass.edu/~casutton/publications/crf-tutorial.pdf. 95

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7*, ConLL '00, pages 127–132, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. 42

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. 43, 120

B. Todorovic, S. Rancic, I. Markovic, E. Mulalic, and V. Ilic. Named entity recognition and classification using context Hidden Markov Model. In *Neural Network Applications in Electrical Engineering, 2008. NEUREL 2008. 9th Symposium on*, pages 43–46, September 2008. 43, 45

A. Tori. *Zemanta service*. Zemanta, 2008. URL `http://developer.zemanta.com/docs/Zemanta_API_companion`. 70

H. Uszkoreit. What is Computational Linguistics, 2000. URL `http://www.coli.uni-saarland.de/~hansu/what_is_cl.html`. 30

W3C. Linked data, 2010a. URL `http://www.w3.org/standards/semanticweb/data`. 61

W3C. Vocabularies, 2010b. URL `http://www.w3.org/standards/semanticweb/ontology`. 58, 59

M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular Value Decomposition and Principal Component Analysis. *ArXiv Physics e-prints*, 8 2002. 90

K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds. Efficient RDF Storage and Retrieval in Jena2. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *SWDB*, pages 131–150, 2003. 100

Y. Wilks. Information Extraction as a Core Language Technology, What is IE? In *SCIE '97: Int. Summer School on IE*, pages 1–9, London, UK, 1997. Springer. ISBN 3-540-63438-X. 35, 49

D. C. Wimalasuriya and D. Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 36(3): 306–323, 2010. 15, 66, 67, 68, 69, 72

R. Yangarber and R. Grishman. Customization of Information Extraction Systems. In *Proceedings of International Workshop on Lexically Driven Information Extraction*, July 1997. 49